Ю.П.Боглаев

Вычислительная математика и программирование

Ю. П. Боглаев

ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ПРОГРАММИРОВАНИЕ

Допущено
Государственным комитетом СССР
по народному образованию
в качестве учебного пособия
для студентов высших технических
учебных заведений

Москва
«Высшая школа»
1990

ББК 22.18
Б73
УДК 681.3.06

Рецензенты: кафедра прикладной математики Обнинского института атомной энергетики (зав. кафедрой — д-р физ.-мат. наук, проф. В. А. Тупчиев) и канд. физ.-мат. наук В. Г. Сушко (Московский государственный университет им. М. В. Ломоносова)

Боглаев Ю. П.

Б73 Вычислительная математика и программирование: Учеб. пособие для студентов втузов.— М.: Высш. шк., 1990.— 544 с.: ил.

ISBN 5-06-000623-9

В книге излагаются сведения, необходимые для проведения научно-технических расчетов на ЭВМ. Рассматриваются архитектура ЭВМ, структура вычислительных алгоритмов и программ, программирование на фортране, системные и инструментальные программы, методы вычислений. Тесная связь программирования и методов вычислений иллюстрируется многочисленными примерами.

Б $\frac{1602120000(4309000000)-435}{001(01)-90}$ 88—90

ББК 22.18
517.8

ISBN 5-06-00623-9

© Ю. П. Боглаев, 1990

ОГЛАВЛЕНИЕ

Предисловие 5
Введение 7

Глава 1
Архитектура ЭВМ

1.1. Введение 15
1.2. Архитектура фон Неймана 16
1.3. Классификация архитектурных решений ЭВМ 22
1.4. Вычислительные системы серии СМ 28
1.5. Память на магнитных дисках и лентах 33
1.6. Алфавитно-цифровые печатающие устройства 37
1.7. Алфавитно-цифровой терминал 38

Глава 2
Структура алгоритмов и программ

2.1. Семейства вычислительных алгоритмов 41
2.2. Структура алгоритмов 46
2.3. Поток данных 52
2.4. Структура программ 54

Глава 3
Программирование на фортране

3.1. Введение 61
3.2. Основы программирования на фортране 63
3.3. Программирование элементарных вычислительных алгоритмов 90
3.4. Оптимизация программ 116
3.5. Расширение возможностей фортрана 123

Глава 4
Системные и инструментальные программы

4.1. Введение 133
4.2. Элементы операционных систем 137
4.3. Система реального времени 143

Глава 5
Методы вычислений

5.1. Примеры 164
5.2. Масштабирование и замена переменных 172
5.3. Аналитические методы 178
5.4. Методы возмущений 224
5.5. Численные методы 235
5.6. Оценка результатов вычислений 239
5.7. Особенности серийных вычислений 246

Глава 6
Теория приближений

6.1. Введение 248
6.2. Интерполяция 251

6.3. Сплайны 256
6.4. Равномерные приближения 261
6.5. Среднеквадратичные приближения 268

Глава 7

Численное интегрирование

7.1. Введение 276
7.2. Простейшие квадратурные формулы 279
7.3. Составные квадратурные формулы 282
7.4. Оценка погрешности численного интегрирования 288
7.5. Формулы Гаусса 293
7.6. Интегрирование функций двух переменных 297

Глава 8

Линейная алгебра. Линейная оптимизация

8.1. Оценки погрешности решения задач линейной алгебры 301
8.2. Прямые методы решения систем линейных уравнений 312
8.3. Итерационные методы решения систем линейных уравнений 322
8.4. Вычисление собственных значений и векторов 328
8.5. Линейная оптимизация 335

Глава 9

Нелинейные уравнения. Нелиинейная оптимизация

9.1. Введение 344
9.2. Сжимающие отображения 348
9.3. Метод Ньютона 356
9.4. Метод бисекции 361
9.5. Поиск минимума функции 363
9.6. Методы спуска 367
9.7. Метод золотого сечения 371

Глава 10

Обыкновенные дифференциальные уравнения

10.1. Дискретизация 375
10.2. Задача Коши. Методы Рунге — Кутта 387
10.3. Жесткие уравнения 396
10.4. Краевые задачи 402

Глава 11

Уравнения с частными производными

11.1. Введение 415
11.2. Разностный метод решения стационарных уравнений 422
11.3. Разностный метод решения нестационарных уравнений 433
11.4. Метод прямых 440

Глава 12

Библиотека фортран-программ

12.1. Каталог библиотеки 445
12.2. Содержание разделов 446
12.3. Описание программ 450

Задачи и упражнения 490

I уровень 490
II уровень 510
III уровень 518

Приложение 1. Англо-русский словарь 525
Приложение 2. Фортран 77. 529
Литература 534
Предметный указатель 536

ПРЕДИСЛОВИЕ

Вычислительная математика и программирование (ВМП) занимают важное место в общеобразовательной подготовке будущего инженера. Соответствующие курсы читаются в различных вариантах во всех втузах страны.

Однако при всем разнообразии методик преподавания можно выделить два принципиально различных направления: 1) вычислительная математика и программирование читаются как отдельные курсы; 2) ВМП — единый курс. При написании книги автор придерживался второго направления. Настоящая книга написана в соответствии с программой курсов «Основы программирования» и «Основы математического моделирования».

Специфика книги состоит в следующем: программирование и вычислительная математика не излагаются как самостоятельные науки с обязательной концентрацией внимания на особенностях каждой дисциплины в отдельности. Главная цель пособия — научить решать на ЭВМ задачи математического моделирования технических процессов. Для достижения этой цели следует кроме программирования и численных методов освоить операционную систему ЭВМ, связать оба предмета общими задачами, научиться пользоваться промышленными библиотеками. При этом некоторые тонкости языка программирования, а также доказательства оценок погрешности, сходимости некоторых алгоритмов вычислительной математики пришлось опустить.

При выборе языка программирования, операционной системы, численных методов, типа ЭВМ в первую очередь учитывались потребности специалистов, которых готовят вузы политехнического профиля. Рассматривается язык высокого уровня — фортран. Изложение архитектуры, операционной системы ведется с ориентацией на применение ЭВМ типа СМ. Выбор конкретного типа машин обусловлен их распространением и возможностью использования в расчетах по математическому моделированию физических, химических, технологических и т. п. процессов. Операционная система реального времени ОС РВ предоставляет возможность решать одновременно задачи управления приборами, технологическими установками, стендами и т. д. Эти задачи могут успешно решаться после изучения машинно-ориентированного языка макроассемблера.

Если курс ВМП организуется на базе другой вычислительной техники, например ЕС, с другой операционной системой, то следует соответственно заменить содержание 1.4—1.7; 4.3. Изучение остального материала книги не зависит от специфики используемых ЭВМ. Численные методы и состав библиотеки фортран-программ отобраны на основе анализа распространенных библиотек для научно-технических расчетов в объеме, достаточном для начального этапа обучения.

Минимальный объем курса ВМП включает содержание глав 3, 4, 6—11 в соответствии с профилем специальности студента. Главы 2 и 5 ориентированы на углубленный подход к предмету, их содержание поможет в проведении типовых расчетов, выполнении курсовых работ и т. п.

Изложение программирования строится на основе алгоритмов численного анализа. Начальным итогом обучения является умение написать самостоятельно программу (по данному алгоритму) и отладить ее на ЭВМ. Далее акцент переносится на использование библиотечных программ и анализ собственно численных методов. Курс завершается решением на ЭВМ задачи технического содержания, которая является типичной для будущей специальности студента.

Автор приносит глубокую благодарность академикам Ю. А. Осипьяну, А. А. Самарскому, А. Н. Тихонову, профессором В. Ф. Бутузову, А. Б. Васильевой, Ю. А. Дубинскому, В. П. Кутепову, С. А. Ломову, Н. Х. Розову, В. А. Тупчиеву, А. Б. Фролову, доцентам А. А. Амосову, Н. В. Копченовой, В. Г. Сушко, чей научно-педагогический опыт был явно и неявно использован в работе над книгой.

Все замеченные недостатки, предложения просьба направлять автору в адрес изд-ва «Высшая школа». 101430, Москва, ГСП-4, Неглинная ул., 29/14.

Автор

ВВЕДЕНИЕ

Во все времена человек стремился расширить свои возможности, создавая разнообразные орудия труда, познания мира, средства существования. Так, например, недостаточность зрениия компенсируют микроскоп, телескоп, радиолокатор. Ограниченные возможности передать информацию друг другу расширяются телефоном, радио, телевидением.

Вычислительные машины «дополняют» возможности человеческого мозга, расширяют его возможности по обработке информации, позволяют увеличить скорость принятия решения в ходе выполнения какой-либо работы.

В конце 40-х годов XX в. работы в области ядерной физики, баллистики управляемых снарядов, аэродинамики и т. д. потребовали такой вычислительной работы, которую уже было невозможно выполнить с помощью арифмометров — основного вычислительного инструмента тех лет. Наука и техника были поставлены перед дилеммой: или всем сесть за арифмометры, или найти новый эффективный инструмент вычислений. Аналогичные проблемы уже не раз возникали и будут еще не раз вставать перед учеными и инженерами: экстенсивный путь развития далее неприемлем, необходим новый, интенсивный путь. Проблема была решена созданием универсальной вычислительной машины. Термин «универсальная» использован не случайно: специализированные машины (например, для обработки банковских счетов, инвентаризации на перфорационных машинах) существовали и ранее, но не было машины, команды для которой, записанные в память, можно было быстро заменить новыми.

Кроме арифметических вычислений ЭВМ может выполнять и логические, т. е. делать выбор между вариантами (ветвями) продолжения действий в зависимости от выполнения некоторых условий. Таким образом, ЭВМ — это нечто большее, чем «быстрый арифмометр».

Краткая характеристика различных поколений ЭВМ.

Первое поколение ЭВМ. Техническая основа элементной базы машин первого поколения — электронные лампы. Максимальное быстродействие $\sim 10^2$ арифметических операций в секунду (оп/с), объем оперативной памяти 10^2 слов. Режим использования — монопольный, т. е. в распоряжении пользователя были все ресурсы машины и ее управление.

Второе поколение ЭВМ. Техническая основа — транзисторы. Максимальное быстродействие $\sim 10^4$ оп/с, объем оперативной памяти $\sim 10^4$ слов, внешняя память на магнитных лентах, дисках $\sim 10^7$ слов. Режим использования — пакетная обработка. Задания для ЭВМ (на перфокартах или магнитных лентах, дисках) собирались в пакет, который обрабатывался без перерыва между заданиями. Такой способ диктовался в первую очередь экономическими соображениями.

Третье поколение ЭВМ. Техническая основа — большие интегральные схемы (БИС), которые на малых полупроводниковых кристаллах реализуют большие схемы машин второго поколения. Максимальное быстродействие $\sim 10^6$ оп/с, оперативная память $\sim 10^6$ слов, внешняя память $\sim 10^9$ слов. Метод использования — режим разделения времени совместно с пакетной обработкой. Высокое быстродействие позволяет время обслуживания пользователей разбить на кванты, обрабатывая в течение кванта задание каждого, возвращаться к пользователю за такое малое время, что у него создается иллюзия за дисплеем, что только он один «владеет» машиной. Положение пользователя стало похожим на монопольное владение ЭВМ первого поколения.

Четвертое поколение ЭВМ. Техническая основа — сверхбольшие интегральные схемы (СБИС); на одном полупроводниковом кристалле реализуются основные элементы ЭВМ третьего поколения, наличие многих параллельно работающих устройств обработки информации. Имеется тесная связь аппаратурной и программной реализаций в структуре машин, наличие элементов новой архитектуры ЭВМ. Традиционная архитектура ЭВМ фон Неймана доминировала на протяжении трех поколений. Максимальное быстродействие $\sim 10^9$ оп/с, оперативная память $\sim 10^7$ слов, внешняя память ограничена в основном экономическими соображениями. Метод работы — доступ к ЭВМ, объединенным в сети; использование пакетов программ с языками программирования, близкими к профессиональным.

Пятое поколение ЭВМ. Проекты ЭВМ пятого поколения находятся в стадии реализации. Но уже из этих проектов и первых опытных экземпляров ЭВМ можно заключить, что максимальное быстродействие арифметических вычислений дополняется здесь высокими скоростями логического вывода. Даже скорость предполагается выражать в единицах лвс (логический вывод в секунду). Форма общения с ЭВМ на естественном языке и дисциплина программирования как наука для пользователя перестают в будущем быть актуальными. Знания, накопленные в различных областях человеческой деятельности, хранятся в памяти ЭВМ в базах знаний, и с их же помощью они расширяются и пополняются. Бумажные носители информации исчезают. Дальнейшее описание проектов было бы похоже на жанр научной фантастики.

Бурное развитие вычислительной техники предопределяется задачами, выдвигаемыми наукой и техникой, и, в свою очередь,

оказывает непосредственное воздействие на области своих приложений. Космическая техника и атомная энергетика, проектирование сложных объектов машиностроения и управление быстропротекающими процессами, создание баз данных и знаний немыслимы без применения ЭВМ. Сфера применения ЭВМ охватывает практически все направления человеческой деятельности и постоянно расширяется.

Рассмотрим два важных направления использования ЭВМ: математическое моделирование и управление процессами. Именно здесь вычислительная математика и программирование имеют фундаментальное значение.

Математическое моделирование. Имеется конкретная техническая задача, ищутся различные подходы к ее решению. Пусть, например, нужно спроектировать новую турбину ГЭС. Чтобы знать поведение турбины, следует решить задачи гидродинамики, механики (динамика, прочность, устойчивость), управления (переход с одного режима на другой) и т. д. В распоряжении специалиста имеются старые технические решения, которые частично можно использовать, возможно пойти по пути создания натуральной модели и, наконец, частично или полностью описать поведение турбины с помощью математической модели.

Если реальный процесс, возможно и стационарный (не зависящий от времени), описан достаточно точно математическими соотношениями, то говорят о построении математической модели данного процесса. Полная схема математического моделирования приведена на рис. В.1.

Для изучения математической модели необходимо разработать вычислительный метод (метод решения), причем лишь в редких случаях удается найти метод, приводящий к точному решению, как правило, используются приближенные решения совместно с точными. Важнейшей задачей, сопутствующей выбору вычислительного метода, является оценка погрешности получаемого решения.

Процесс решения задачи согласно выбранному вычислительному методу описывается в виде *алгоритма*. На интуитивном уровне алгоритм определяется как процесс построения величин, идущий в дискретном времени, который позволяет из системы величин в предыдущий момент времени получить систему величин

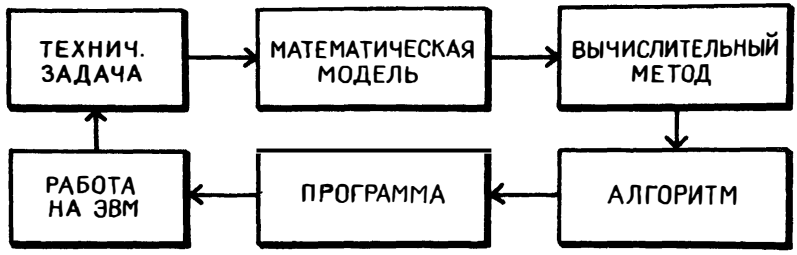

Рис. В.1

в последующий момент, для которого задается начальная система и сформулировано правило окончания процесса. Примерами могут служить известные алгоритмы Евклида нахождения наибольшего общего делителя двух положительных натуральных чисел и алгоритм Гаусса решения системы линейных уравнений методом исключения, а также алгоритмы игры в шахматы, перевода с одного языка на другой, сортировки и поиска в массивах информации.

Чтобы алгоритм был реализован на ЭВМ, необходимо написать соответствующую этому алгоритму программу. *Программа — это совокупность команд, которые может выполнить машина.* Программирование и состоит в работе по подготовке этой совокупности команд.

В ЭВМ первого поколения эта работа целиком выполнялась пользователем, который, зная систему команд ЭВМ (коды команд) и доступные ему ресурсы, готовил программу в кодах — на языке машинных команд. Программу на языке машинных команд ЭВМ, размещенную в памяти, машина может выполнить сразу же после команды запуска ее на счет. Однако программирование в машинных кодах — очень трудоемкий процесс. Для упрощения программирования были разработаны языки программирования более высокого уровня — ассемблер, алгол, фортран, бэйсик, паскаль и т. д. Программирование на языке высокого уровня, упрощая процесс подготовки программы, требует применения специальной программы (принадлежащей математическому обеспечению ЭВМ) — транслятора с языка высокого уровня. Аналогичные процедуры имеют языки с интерпретаторами и ЭВМ с аппаратно реализованными трансляторами.

Таким образом, составление программы, готовой к исполнению, включает применение промышленного математического обеспечения ЭВМ, в частности операционной системы (ОС), и умение с ней работать.

«Стиль» программирования за время развития этой дисциплины значительно изменился. Программирование в задачах технического характера все более походит на конструирование программ из готовых программных модулей. За прошедшие годы накоплены обширные библиотеки программ как общего применения, так и специального, организованные в пакеты прикладных программ. В этой связи повышается внимание к возможности переноса программы с одной машины на другую, к формированию личных профессиональных библиотек пользователя, а в итоге к концентрации усилий инженера на главной цели — решение научно-технической задачи.

Последним этапом работы на ЭВМ является запуск программы на счет и получение числовых результатов. Затем пользователь должен соотнести их с поставленной технической задачей и, возможно, изменить модель, метод, алгоритм, программу. Цикл, изображенный на рис. В.1, повторяется до тех пор, пока цель не будет достигнута.

Проиллюстрируем описанную схему математического моделирования на примере следующей технической задачи. Найти на

интервале времени $t_0 \leqslant t \leqslant t_n$ переходный процесс в системе управления турбиной, описываемой вектором состояния $x(x_1, ..., x_{10})$ с заданным начальным вектором

$$x(t_0) = x^{(0)}. \tag{1}$$

Математическая модель. Предположим, что поведение вектора состояния во времени можно описать обыкновенным дифференциальным уравнением

$$\frac{dx}{dt} = f(x, t). \tag{2}$$

Чтобы получить уравнение (2), необходимо использовать методы теории управления, механики, выделить главные черты управляемого движения и пренебречь второстепенными. Соотношения (1) и (2) и являются математической моделью процесса управления.

Вычислительный метод. Точное решение (1), (2) удается найти в исключительных случаях (линейные уравнения с постоянными коэффициентами, переменные разделяются, специальные уравнения). Как правило, используют приближенные методы. Простейший из них — явный метод Эйлера. Интервал $[t_0, t_n]$ покрывается дискретным множеством $t_i = \{t_0 + ih\}$, где h — шаг интегрирования, $i = 0, 1, 2, ..., N$. Будем искать приближения x_i к вектор-функции $x(t)$ на множестве t_i, следуя методу Эйлера:

$$x_{i+1} = x_i + hf(x_i, t_i), \quad x_0 = x^{(0)}. \tag{3}$$

Формулы (1), (3) представляют собой вычислительный метод, который может дать некоторые приближения x_i, $i = 1, 2, ..., N$, к точным значениям $x(t_i)$ на множестве t_i. Важнейшей задачей выбора вычислительного метода является обеспечение сходимости приближений к точному решению $(\|x_i - x(t_i)\| \to 0, \ h \to 0)$ и оценка погрешности приближений $(\|x_i - x(t_i)\| = ?)$.

Алгоритм. Алгоритм метода Эйлера без контроля точности можно описать следующим образом.

1. Задать числа t_0, t_N, h, вектор x_0.
2. Вычислить $N = [(t_N - t_0)/h]$.
3. По вектору x_i вычислить вектор x_{i+1}, следуя формуле (3), и выдать его значение на дисплей.
4. Закончить процесс вычислений после определения вектора x_N.

Программа. Возможный вариант программы на фортране приведен ниже. Перевод английских слов можно найти в словаре (см. Приложение 1):

```
      REAL X(10),F(10),TO,
      REAL TN,H,X
      INTEGER N,I,J
      READ (5,1) TO,TN,H,X
1     FORMAT (F7.4)
      N=(TN-TO)/H
      DO 3 I=1,N
```

```
      SUBROUTINE S(F,X,T)
      REAL X(10),F(10),T
      F(1)=T+X(2)**2
      F(2)=X(3)-X(1)
      .....
      .....
      F(10)=COS(X(9))
```

```
      CALL S(F,X,TO)              RETURN
      DO 2 J=1,10                 END
2     X(J)=X(J)+H*F(J)
      WRITE (5,1) X
3     TO=TO+H
      END
```

В правом столбце представлена часть программы, вычисляющая правую часть (2):

$$f_1 = t + x_2^2,\ f_2 = x_3 - x_1,\ \ldots,\ f_{10} = \cos x_9.$$

Работа на ЭВМ. Пользователь ЭВМ должен быть зарегистрирован в ОС РВ администратором, ему присваивается код идентификации, например [231, 1], он должен иметь пароль, например TOR. Работа на ЭВМ заключается в наборе следующих команд на клавиатуре дисплея (каждая строка заканчивается нажатием клавиши возврата каретки). Подчеркнутые символы выдаются на экран дисплея ОС, пропущенный текст заменен многоточием:

```
> HELLO
> [231,1]
> PASS WARD: TOR
.....
.....
> EDI UPR,FTN
.....
INPUT
набор текста программы, приведенного выше
.....
два нажатия возврата каретки
* EX
> FOR UPR,UPR=UPR
> TKB UPR/CP=UPR,[1,1] FOROTS/LB
> RUN UPR
набор чисел TO,TN,H, вектора X
.....
выдача на дисплей результатов вычислений
> TT: - - STOP
> BYE
```

Здесь приведен вариант дисплейного сеанса от входа в ОС по команде HELLO (здравствуйте) до выхода из ОС по команде BYE (до свидания) в предположении, что пользователь не совершал ошибок. Реально работа на ЭВМ состоит в устранении различных ошибок в программе (отладка программы) с помощью диагностики ошибок, которая выдается на экран дисплея программами ОС.

Укажем те элементы математического моделирования, которые находятся в центре внимания настоящей книги. Это вычислительный

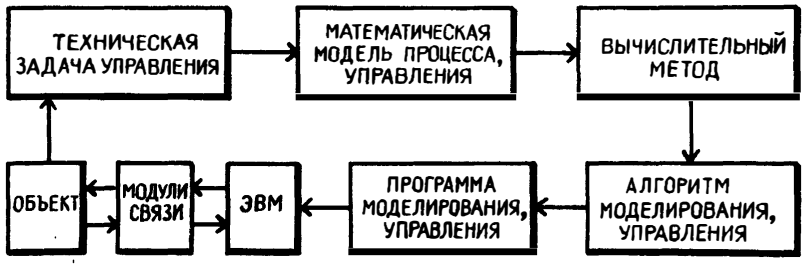

Рис. В.2

метод, алгоритм, программа, работа на ЭВМ. Частично затрагиваются вопросы, связанные с математической моделью, но лишь в аспекте выбора той модели, которая может быть оптимально реализована на ЭВМ, технические задачи приводятся для иллюстраций и пояснения происхождения математических моделей.

Схема рис. В.1 представляет собой способ применения ЭВМ изолированно, в автономном режиме (OFF LINE) от какого-либо реального процесса, который модель описывает.

Наиболее общая форма применения ЭВМ в технических задачах — это одновременное использование ЭВМ для математического моделирования реального процесса, объекта и управление этим процессом, объектом в реальном времени — режим связи ЭВМ с объектом (ON LINE).

ЭВМ в системе управления процессом. Применение ЭВМ для управления технологическими, экспериментальными и т. п. процессами, установками подразумевает наличие модулей связи между объектом управления и ЭВМ. Это объясняется тем, что информация с объекта поступает в аналоговом виде (сила тока, напряжение и т. д.), а ЭВМ обрабатывает информацию, представленную в цифровом виде. Сигналы управления из ЭВМ на объект должны быть преобразованы из цифровой формы в аналоговую. Программа моделирования и управления размещается в памяти машины. Схема управления изображена на рис. В.2. Стрелками показан путь реализации задачи управления и моделирования. Здесь необходимо программировать работу модулей связи, а также учитывать время исполнения различных фрагментов программы на ЭВМ. Важно, чтобы временные характеристики объекта были согласованы с временем исполнения программы. Программа управления переходным процессом не должна работать дольше, чем длится сам этот процесс.

Программирование в реальном масштабе времени работы модулей связи, как правило, ведется на машинно-ориентированных языках типа ассемблера, которое требует более глубокого знания возможностей ЭВМ, ее аппаратуры и ОС, чем программирование задач моделирования на фортране. Однако нельзя утверждать, что ассемблер вообще лучше фортрана, так как каждый язык имеет свой круг алгоритмов, для которых он наиболее удобен. Для программирования задач в реальном масштабе времени

идеальным является владение по крайней мере двумя языками типа ассемблера и фортрана.

Структура книги. Содержание книги можно условно разбить на две части. В первой части (гл. 1—4 и 12) излагаются архитектура ЭВМ, работа за дисплеем, программирование, ОС, библиотека фортран-программ. Во второй части (гл. 5—11) представлены вычислительные методы. Каждая глава этой части независима. Предполагается, что сведения из курса высшей математики, лежащие в основе этих глав, читателю уже известны.

Советы по применению курса ВМП на практике

1. Задачи рекомендуется решать не только в дисплейном классе на ЭВМ коллективного пользования, но и на персональной вычислительной технике (персональные ЭВМ, программируемые микрокалькуляторы).

2. При постановке технической задачи прежде всего следует ознакомиться со всеми методами ее решения на ЭВМ (модели, алгоритмы, программы), известными в данный момент, например в фондах алгоритмов и программ.

3. «Ближайшая» решенная задача, как правило, служит хорошим тестом для проверки решения поставленной задачи, а ее решение (аналитическое или численное) может быть использовано в вычислениях (см. **5.1—5.4**).

4. Решение задачи на ЭВМ, полученное без оценки погрешности, не имеет практического смысла.

5. До начала решения задачи следует оценить возможности ЭВМ: точность представления чисел должна быть достаточной для ее решения, объем памяти должен соответствовать задаче, время вычислений не должно превышать требуемый предел, средства ввода — вывода информации должны обеспечивать потребности задачи.

6. Сведения, выходящие за рамки настоящей книги, можно найти: по архитектуре ЭВМ — [5, 28]; по структуре алгоритмов и программ — [1, 17, 28]; по операционным системам — [5, 7, 12, 14]; по программированию на фортране — [3, 6, 13, 15, 17, 32]; по методам вычислений — [2, 4, 8—11, 15, 16, 18—25, 29—33]; тексты фортран-программ — [3, 6, 9, 13, 15, 26, 27, 30, 32].

Глава 1

АРХИТЕКТУРА ЭВМ

● 1.1. Введение

1.1.1. Пользователь и ЭВМ. На вычислительную машину, как и на любой объект, различные специалисты смотрят по-разному. Конструктор вычислительной техники, инженер-электронщик, видит в ЭВМ набор электронных узлов (плат, интегральных схем, кристаллов) и технологию их изготовления. Разработчика программного обеспечения, инженера-программиста, интересует набор команд, выполняемый ЭВМ, временные характеристики всех устройств. Эксплуатирующие ЭВМ инженеры-электронщики и программисты работают на ЭВМ совершенно в другой манере, нежели разработчики, поскольку они имеют дело с готовым объектом.

Разработчики и специалисты по эксплуатации являются профессионалами в вычислительной технике, для них ЭВМ — основной объект приложения профессиональных знаний.

В то же время имеется большая группа людей, пользователей ЭВМ, для которых вычислительная техника — инструмент решения разнообразных задач (учебных, научно-технических, информационных и т. п.). Естественно, что внутри этих групп: разработчиков, специалистов по эксплуатации, пользователей — есть своя иерархия (тонкая структура), которую легко понять поработав в этой группе.

Отмеченные три группы специалистов возникают практически во всех технических областях человеческой деятельности. Например, автомобиль раньше, чем ЭВМ, также «породил» три группы связанных с ним людей: разработчиков, специалистов по эксплуатации и водителей.

Аналогия между водителями и пользователями поможет выяснить состав этих групп. Водители не однородны по своему составу; имеются профессиональные водители и автолюбители, гонщики-спортсмены как среди профессионалов, так и среди любителей, спортсмены, в свою очередь, также не однородны (различные виды машин, виды гонок), наконец, есть каскадеры. Аналогичную неоднородность состава представляет группа пользователей: «каскадеры», виртуозно владеющие ЭВМ; пользователи, выжимающие из машины больше, чем заложено ее создателями (спортсмены), пользователи-профессионалы и рядовые пользователи (автолюбители).

Очевидно, что водить автомобиль можно зная только правила дорожного движения и назначение трех педалей, руля и рычага переключения передач. Точно так же можно пользоваться ЭВМ зная элементы какого-нибудь языка программирования, назначение клавиш терминала и несколько команд операционной системы. Однако такой уровень владения техникой приводит пользователя (водителя) в беспомощное состояние из-за малейших пустяков. Недаром, чтобы получить права, следует сдать экзамены по материальной части автомобиля.

В настоящей главе рассматривается «материальная часть» ЭВМ, называемая архитектурой.

1.1.2. Архитектура ЭВМ. Под архитектурой машины понимается ее логическая организация, структура, ресурсы, которые может использовать программист. Описание ЭВМ в виде логических элементов (а не физических) и их взаимодействия друг с другом освобождает пользователя от необходимости знания физической организации элементов ЭВМ.

Иметь представление об архитектуре полезно, а часто и необходимо пользователю на любом этапе вычислительного процесса от формулировки математической модели до проведения вычислений. Архитектура определяет те ресурсы, которыми располагает пользователь для решения своей задачи. Возможно, что на этапе формулировки модели окажется, что архитектура ЭВМ не обеспечивает решение поставленной задачи, и тогда для ее решения следует выбрать машину подходящей архитектуры.

● 1.2. Архитектура фон Неймана

1.2.1. Основные элементы. Структура, присущая большинству современных ЭВМ, была принципиально описана в конце 40-х годов фон Нейманом. Ее иногда называют классической или традиционной. Общая организация ЭВМ представлена на рис. 1.1.

Центральное место среди функций, выполняемых машиной, занимают арифметические и логические операции. Для этой цели служит блок, называемый арифметическим и логическим устрой-

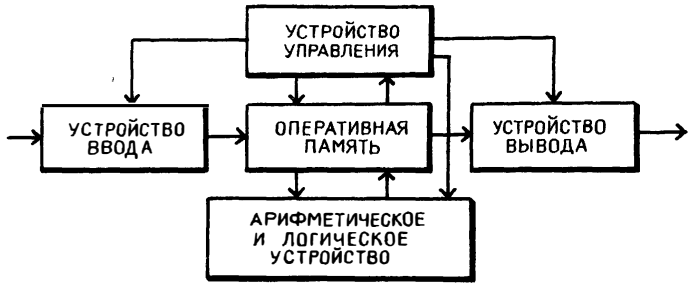

Рис. 1.1

ством. Второй блок — устройство управления. Этот блок обеспечивает последовательность управляющих сигналов для арифметического и логического устройства.

При выполнении арифметических операций возникает необходимость в сохранении промежуточных результатов. Роль хранилища информации, доступной как арифметическому и логическому устройству, так и устройству управления, выполняет оперативное запоминающее устройство или память.

Для того чтобы использовать машину для обработки необходимой информации и получения результатов, нужно вводить и выводить информацию. Эти операции обеспечивают устройства ввода и вывода (рис. 1.1).

1.2.2. Арифметическое и логическое устройство. Это устройство в ЭВМ фактически и выполняет все вычисления. Операции выполняются с помощью электронных схем, каждая из которых состоит из нескольких тысяч элементов. Микросхемы имеют высокую плотность и быстродействие. На современном технологическом уровне все арифметическое и логическое устройство можно разместить на одном кристалле полупроводникового элемента размером с конторскую скрепку.

1.2.3. Устройство управления. Этот блок управляет работой всей ЭВМ. Здесь оценивается поток входной информации и решается вопрос: когда, как и какими средствами выполнять обработку. Устройство управления указывает арифметическому и логическому устройству последовательность работ, источник получения информации, а также приемник результатов.

Управляющие сигналы вырабатываются на основе интерпретации последовательности команд. Команда содержит информацию о том, что нужно делать (код операции) и где расположены данные (адреса операндов в памяти). Последовательность команд для устройства управления хранится в памяти и называется собственно программой. Это программа в машинных кодах.

Команды выполняются последовательно во времени в порядке, который определяет либо сама команда, либо специальный счетчик команд. В некоторых машинах совмещается во времени выполнение текущей команды с выборкой из памяти следующей. Но эта процедура «не видна» программисту, она выполняется на электронном уровне.

Устройство управления совместно с арифметическим и логическим устройством называют центральным процессором.

1.2.4. Оперативная память. Этот блок хранит информацию для устройства управления (команды) и арифметического устройства (данные). Идея размещения в памяти команд и данных является фундаментальной, лежащей в основе ЭВМ архитектуры фон Неймана. Содержимое ячейки памяти, на которую указывает счетчик команд, может интерпретироваться как команда или как данные.

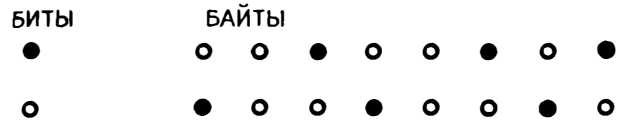

Рис. 1.2

Оперативная память собирается на ферритовых сердечниках или полупроводниковых микросхемах и состоит из отдельных ячеек. Каждая ячейка может содержать элемент данных (числа), часть команды или полную команду.

Простейшей единицей измерения данных является двоичный разряд — *бит*. Бит имеет представление «0» или «1» и запоминается как определенное направление намагниченности ферритового сердечника либо определенное состояние электронного переключателя. Отдельный бит обрабатывается и изменяется только в центральном процессоре.

Многие ЭВМ имеют байтовую архитектуру. *Байт* — это последовательность битов, образующих единицу передачи информации. Обычно байт состоит из 8 бит (рис. 1.2). На ЭВМ с байтовой архитектурой предусматривается обращение к отдельным байтам.

Упорядоченная совокупность битов (байтов) образуют слово. Количество битов в слове изменяется от 8 (для микро-ЭВМ) до 64 (в супер-ЭВМ). В машинах с байтовой архитектурой слово представляется 2, 4, 8 байтами. Машинное слово — это основной объект данных в языке фортран (см. гл. 3).

Каждое слово (ячейка) имеет свой адрес, связанный с расположением в памяти. Следует отличать адрес от содержимого (рис. 1.3).

1.2.5. Организация памяти. В любой ЭВМ, даже в калькуляторе, существует несколько уровней памяти. Важной особенностью иерархии уровней является обратно пропорциональная зависимость емкости от стоимости (скорости доступа). Чем больше емкость памяти, тем медленнее к ней доступ (рис. 1.4). Время доступа — это время, необходимое для выборки из памяти или записи в нее информации. Характеристики шести уравнений памяти приведены в табл. 1.1.

Самую быструю часть памяти представляет собой часть центрального процессора — регистры. Регистры образуют два набора: регистры команд и регистры данных. В зависимости от характера данных регистры организованы в группы. Имеются адресные регистры и регистры данных различных типов.

Рис. 1.3

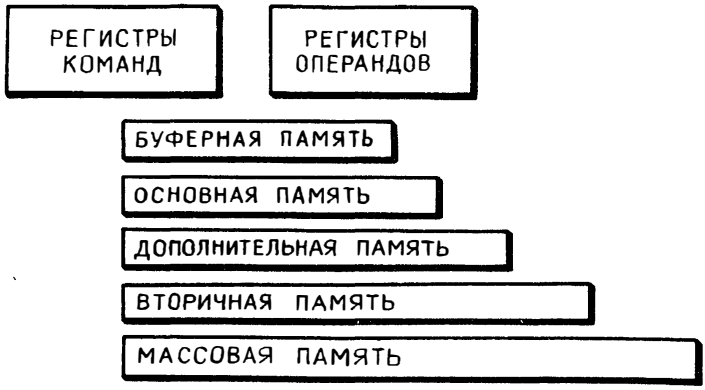

Рис. 1.4

Таблица 1.1

Тип памяти	Время доступа, с	Емкость, бит
Регистры	$(2 \div 20) \cdot 10^{-9}$	$10^3 — 10^4$
Буферная	$(20 \div 200) \cdot 10^{-9}$	$10^4 — 10^5$
Основная	$(0{,}2 \div 20) \cdot 10^{-6}$	$10^6 — 10^8$
Дополнительная	$(1 \div 10) \cdot 10^{-3}$	$10^7 — 10^9$
Вторичная	$(20 \div 100) \cdot 10^{-3}$	$10^9 — 10^{11}$
Массовая	$10 \div 100$	$10^{11} — 10^{12}$

Данные и команды поступают в центральный процессор со следующего уровня, доступного только через устройство управления. Этот уровень называется буферной памятью (кэш-памятью). Он служит для согласования скорости процессора и основной памяти, быстродействие буфера почти такое же, как и регистров, емкость значительно меньше основной памяти. Использование буфера обычно недоступно пользователю, и возникает иллюзия, что основная память работает почти со скоростью процессора. Техническое решение, лежащее в основе буферной памяти, известно как «предварительно просматриваемая память».

Основная (или оперативная) память расширяется двумя типами памяти: дополнительной и вторичной, реализуемой обычно на магнитных дисках. Диски дополнительной памяти имеют фиксированные головки чтения-записи, а вторичной памяти — подвижные головки.

Программа пользователя может обращаться к любой ячейке дополнительной памяти в диапазоне адресов, во много раз превышающем емкость основной памяти. Проведение всех операций, необходимых для вызова информации в основную память, которая требуется центральному процессору или обратной пересылке, выполняется операционной системой (гл. 4). Пользователь вообще не знает, где физически расположена ячейка с данным адресом. Такой адрес памяти называется виртуальным, а организация памяти — виртуальной памятью. Истинный физический адрес

каждого слова данных из виртуального адресного пространства известен только операционной системе.

Данные виртуальной памяти пересылаются из основной в дополнительную память и обратно, согласно специальным алгоритмам, порциями — страницами. В основе этих алгоритмов лежит идея сохранения в основном памяти только тех страниц, на которые наиболее часто ссылается центральный процессор.

Вторичная память требует от пользователя употребления специальных команд ввода — вывода, имеющихся в языках программирования, например фортране (гл. 3).

Наконец, массовая память предназначена обычно для архивных целей, реализуется на магнитных лентах, оптических дисках, микрофильмах и т. п.

Различные виды памяти применяются только из-за неодинаковой стоимости устройств хранения информации. Если бы устройства с минимальным временем доступа были достаточно дешевыми, то следовало бы использовать только один уровень памяти. Из-за отсутствия в настоящее время быстродействующей дешевой памяти приходится сочетать дорогие устройства с дешевыми для обеспечения заданного быстродействия ЭВМ.

1.2.6. Центральный процессор. Как уже отмечалось, в центральном процессоре, объединяющем арифметические и логические устройства с устройством управления, выполняются команды или, что то же самое, программа пользователя. Центральный процессор представляет собой достаточно сложное электронное устройство, состоящее из нескольких функциональных блоков (например, блока сложения, блока умножения и т. п.).

Внутри функционального блока управление выполнением соответствующих операций может производиться аппаратно (электронной схемой) либо программой (микропрограммно). Во втором случае управление блоком осуществляет программа, действующая на *биты операндов* (операнд-объект, над которым производится операция), что и приводит к выполнению операции, например умножения (операнды-сомножители). Эти программы пишут разработчики-программисты. Технология микропрограммного управления позволяет легко увеличивать скорость работы соответствующего блока благодаря применению более совершенных (коротких) программ без переделок аппаратуры.

Современные ЭВМ могут выполнять сотни команд, которые все вместе образуют систему команд. Максимальное количество команд в данной машине ограничивается размером поля, отводимого под код операции. Если это поле занимает 1 байт, то возможны $2^8 = 256$ команд. Система команд полностью отражает архитектуру ЭВМ, поскольку однозначно определяет работу центрального процессора. Остальные части ЭВМ — память, устройства ввода — вывода — должны обеспечивать работу центрального процессора: снабжать командами, операндами, сохранять результаты. Причем при решении тех задач, для которых в основном

применяется ЭВМ, центральный процессор должен активно работать.

Последнее очень важно для оценки производительности процессора и ЭВМ в целом, поскольку в задачах различно сочетание операций, необходимых для их решения. Например, в ЭВМ, применяемой для научно-технических расчетов, желательно иметь более быстродействующие арифметические блоки, нежели в ЭВМ для экономических расчетов. Иллюстрацией может служить типичный набор команд (подробно представленные команды изложены в гл. 3) для двух классов задач:

Научно-техническая задача	*Экономическая задача*
15 умножений	5 умножений
12 делений	2 деления
25 сложений	25 сложений
22 вычитания	18 вычитаний
12 записей в память	14 записей в память
2 ввода	8 вводов
2 команды печати	8 команд печати
8 условных переходов	14 условных переходов
2 безусловного перехода	6 безусловных переходов
100 команд	100 команд

Несмотря на большое число команд ЭВМ, их можно разделить на четыре основных типа: арифметические и логические; команды управления; команды внутренней пересылки; команды ввода — вывода.

Команды, поступающие в центральный процессор, содержат следующую информацию:

1. Код операции.
2. Адреса операндов и результата.
3. Адрес следующей команды.

1.2.7. Архитектура адресации. В этом пункте рассматриваются различные архитектуры ЭВМ, связанные с формой адресации. Поставленной задаче (иметь в команде информацию о коде операции и адресах) удовлетворяет ЭВМ с четырехадресным форматом команды (рис. 1.5). Пусть код операции (символический):

сложения ADD
вычитания SUB
умножения MUL

X, Y, Z, W — символические имена ячеек памяти. Тогда команда

MUL X,Y,Z,W

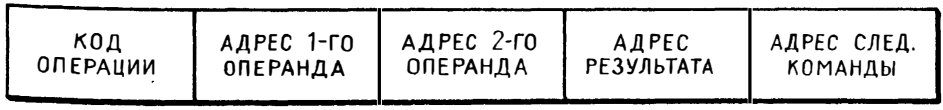

Рис. 1.5

означает: умножить содержимое ячейки X на содержимое ячейки Y, результат поместить в ячейку Z.

В большинстве машин адрес W не указывается, в них имеется указатель — счетчик команд (PC) — это регистр, хранящий адрес текущей команды. В нем во время выполнения текущей команды формируется адрес следующей команды.

Трехадресный формат команды

```
ADD  A,B,C
```

означает: сложить содержимое ячеек A и B, результат поместить в C, а следующую команду взять по адресу, который содержится в регистре PC.

Ради экономии размера ячейки памяти можно пожертвовать одним адресом в формате команды, адрес результата совместить с адресом второго операнда и получить двухадресную машину. Однако, чтобы выполнить сложение C=A+B, теперь потребуется лишняя команда MOV — перемещение содержимого первой ячейки во вторую, а именно:

```
MOV  A,C ;   (A)→(C)
ADD  B,C ;   (B)+(C)→(C)
```

Двухадресные машины широко распространены, такой архитектурой адресации обладают и малые, и большие ЭВМ.

Чтобы получить одноадресную машину, необходимо предусмотреть в ЭВМ еще один регистр — аккумулятор (AC), где будет содержаться второй операнд для арифметических операций и потребуются команды записи в аккумулятор LAC и чтения DAC. Сложение C=A+B в одноадресной машине имеет вид

```
LAC  A ;   (A)→(AC)
ADD  B ;   (AC)+(B)→(AC)
DAC  C ;   (AC)→C
```

Одноадресные машины также широко представлены, а идея иметь аккумуляторы на регистрах используется в двухадресных и трехадресных машинах. Такие регистры, применяемые не только в качестве аккумуляторов, но и для индексации элементов массивов, вычисления числа повторений циклов и т. д., называются регистрами общего назначения.

Использование регистров для хранения операндов и команд, выполняющих операции с их содержимым, значительно повышает быстродействие ЭВМ.

● 1.3. Классификация архитектурных решений ЭВМ

1.3.1. Идея параллельных вычислений. Со времени появления ЭВМ в 40-е годы быстродействие процессора увеличилось на выполнение одной операции примерно с 10^{-1} с до 10^{-8}—10^{-9} с. За время 10^{-9} с электрический сигнал (свет) распространяется на

30 см. Поэтому дальнейшее повышение быстродействия ограничено физическими процессами передачи сигналов и технологией. Нельзя в настоящее время рассчитывать на уменьшение времени срабатывания электронной схемы как на главный резерв ускорения работы процессора. А увеличивать быстродействие ЭВМ постоянно требуют практические задачи. При этом проявляется закономерность, состоящая в том, что для решения необходимых в данное время задач следует иметь ЭВМ, быстродействие которых в 10—100 раз выше, чем у имеющихся машин.

Основной источник задач, требующих быстродействия 10^{-9} с и выше,— управление сложными объектами в реальном масштабе времени, прогноз погоды и др. В таких задачах требуется провести огромный объем вычислений за ограниченный, заданный интервал времени.

Одно из важных направлений увеличения быстродействия — проведение на ЭВМ параллельных вычислений для решения одной задачи. Эта идея высказывалась еще в XIX в., а в ЭВМ она имела некоторое начальное воплощение в рамках традиционной архитектуры фон Неймана. Так, в некоторые процессоры были введены блоки предварительной выборки команд, выполнение команд с перекрытием по времени и т. п. Но широкое развитие идеи параллельных вычислений потребовало разработки принципиально новых архитектурных решений, которые начали реализовываться в действующих ЭВМ в 70-е годы.

1.3.2. Классификация ЭВМ. Любую ЭВМ можно представлять себе как устройство, через которое организовано прохождение двух потоков: потока данных и потока команд. С этой точки зрения практически все новые архитектурные решения и классическую архитектуру определяет табл. 1.2.

В ЭВМ классической архитектуры имеется одно арифметико-логическое устройство (АЛУ), через которое проходит поток данных, и одно устройство управления (УУ), через которое проходит поток команд. Это однопроцессорная ЭВМ.

Наличие в ЭВМ нескольких (Р) процессоров, т. е. объединение АЛУ и УУ, означает, что параллельно может быть организовано много (Р) потоков данных и много (Р) потоков команд. Таким образом, параллельно могут выполняться несколько фрагментов одной задачи. Это многопроцессорная ЭВМ. Структура такой

Таблица 1.2

Поток данных в ЭВМ / Поток команд в ЭВМ	Одиночный	Множественный
Одиночный	Однопроцессорная ЭВМ	Параллельный процессор
Множественный	Конвейерный процессор	Многопроцессорная ЭВМ

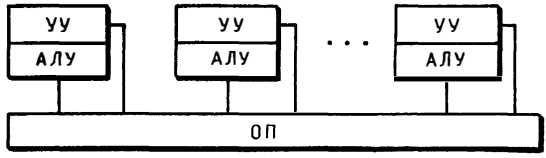

Рис. 1.6

машины, имеющей общую оперативную память (ОП) и несколько процессоров, представлена на рис. 1.6.

Если несколько процессоров, входящих в вычислительную систему, не имеют общей оперативной памяти, а имеют каждый свою (локальную), то это многомашинная вычислительная система. Каждая ЭВМ в многомашинной системе имеет классическую архитектуру, и такая система применяется достаточно широко. Однако задача, решаемая на такой вычислительной системе, должна иметь очень специальную структуру: она должна разбиваться на столько слабо связанных подзадач, сколько ЭВМ в системе. Заметим, что если заботиться не только об увеличении быстродействия решения задач, но и о надежности вычислений, то преимущество многопроцессорных и даже многомашинных вычислительных систем перед однопроцессорными очевидно.

Работа нескольких АЛУ под управлением одного УУ означает, что множество данных может обрабатываться по одной программе — одному потоку команд (фактически это параллельный процессор). Из четырех классов архитектурных решений это наиболее специализированный тип ЭВМ. Высокое быстродействие такой архитектуры можно получить только на задачах, которые удовлетворяют определенным требованиям, а именно:

1) в задаче имеется множественность наборов данных;

2) одинаковые вычислительные операции необходимо выполнить на всех наборах;

3) отсутствие непредвиденных ситуаций в потоках данных.

Структура ЭВМ подобного вида представлена на рис. 1.7.

Следующий тип организации параллельных вычислений идеологически тесно связан с конвейерным способом проведения сборочных работ на производстве. Поясним аналогию на примере. Автомобиль собирают на конвейере 120 рабочих за 60 мин: первый устанавливает шасси, второй закручивает 16 болтов, ..., 120-й отгоняет готовый автомобиль. Хотя сборка каждого автомобиля занимает 60 мин, каждые 30 с с конвейера сходит готовый автомобиль (после загрузки конвейера). Коэффициент ускорения может быть определен отношением 60 мин к 30 с, равным 120, т. е. числу ступеней конвейера.

Аналогично работает вычислительный конвейер. Например, блок сложения двух чисел, представленных в форме с мантиссой и порядком $A=0{,}375\cdot 10^{1}$; $B=0{,}124\cdot 10^{-1}$, можно разбить на пять ступеней:

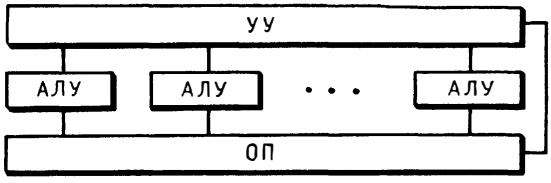

Рис. 1.7

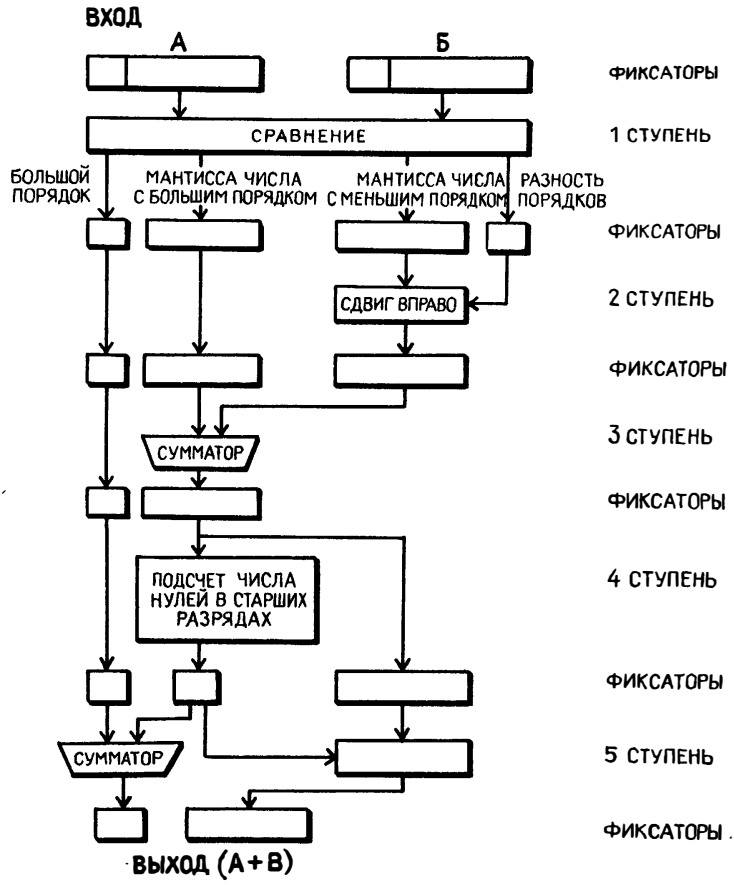

Рис. 1.8

Вход: A, B.

1. Вычитание порядков: $1-(-1)=2$.

2. Сдвиг вправо мантиссы числа, имеющего меньший порядок на величину разности порядков: $0{,}00124\cdot 10^1$.

3. Сложение мантисс:

$$\begin{array}{r} 375 \\ + \\ \underline{001} \\ 376 \end{array}$$

4. Подсчет числа нулей в старших разрядах суммы: 0.

5. Сдвиг влево суммы на число нулей в старших разрядах с изменением порядка (нормализация результата): $0{,}376\cdot 10^1$.

Выход: $A+B$.

Этот блок обычно реализуется на электронном уровне по схеме рис. 1.8, где фиксаторами обозначены элементы, препятствующие перескоку данных с одной ступени на другую, который может произойти из-за различия в количестве логических операций на каждой ступени или изменений временных характеристик логических элементов.

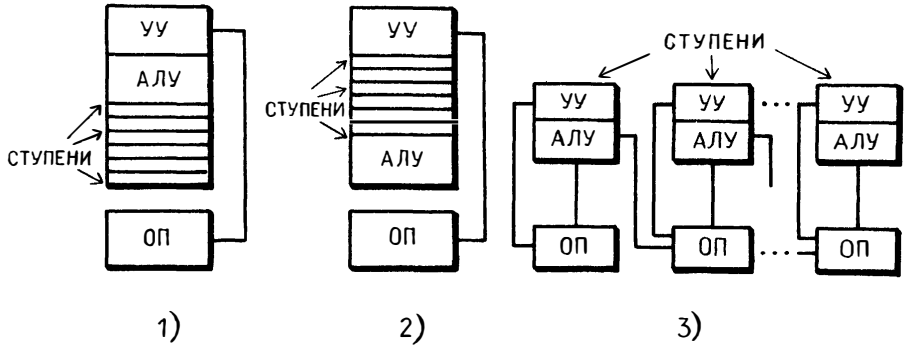

Рис. 1.9

Рассмотренный пример — так называемая аппаратная реализация арифметического конвейера.

Но как параллелизм, так и конвейерную организацию вычислений можно рассматривать на различных уровнях, например:

1) уровень машинных слов: арифметический конвейер;

2) уровень команд: конвейер команд;

3) уровень программ: макроконвейер.

Отмеченные формы конвейерной обработки изображены в виде схем на рис. 1.9. Под конвейерным процессором, помещенным в табл. 1.2, понимается любое вычислительное устройство, имеющее в своей структуре какую-либо конвейерную организацию 1—3.

Ускорение решения задач на вычислительном конвейере определяется числом ступеней, а также характером решаемых задач. Как правило, задачи должны обладать следующими свойствами.

1. Решение задачи не зависит от предыдущих вычислений.

2. Вычисления могут быть представлены в виде одной и той же цепочки подзадач.

3. Подзадачи тесно связаны.

4. Время вычисления подзадач приблизительно одинаково.

Конечно, приведенная классификация ЭВМ в табл. 1.2 слишком «груба» и в современных машинах присутствуют элементы всех четырех типов архитектурных решений. Но основные черты, доминирующие элементы такая классификация отражает.

1.3.3. Архитектура ЭВМ на основе потока данных. В вычислительных системах, рассмотренных выше, порядок вычислений задается последовательностью команд. Этот порядок строго выдерживается при выполнении программы на ЭВМ любой организации, управляемой потоком команд. В 80-е годы была практически реализована новая архитектурная идея управления ходом вычислений — управление потоком данных.

В системах с управлением потоком данных вычисления упорядочиваются в соответствии с готовностью операндов для вычислений. Программа на основе потока данных не является последовательным перечнем команд. Она может быть представлена блок-схемой, где в блоках указываются операции и связи (дуги), по

которым могут перемещаться значения, называемые фишками. Например, фрагмент программы на основе потока данных для вычисления по алгоритму

ввод A,B,C,Y
X=A*Y+B
Y=X/(C*Y+B)

приведен на рис. 1.10.

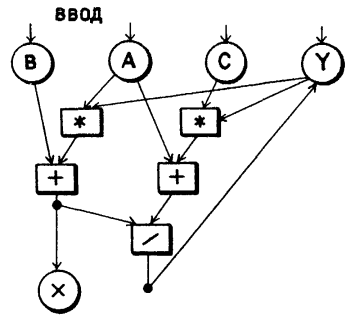

Рис. 1.10

Фишка, которая должна быть подана на несколько блоков, размножается. Элементарная операция может выполняться в любой момент при поступлении на ее входы фишек. Операция поглощает входные фишки, и на выходе появляется фишка, соответствующая выходной дуге операции.

В ЭВМ на основе потока данных «автоматически» используется внутренний параллелизм задачи, который в машинах с управлением потоком команд скрывается в навязанной последовательности команд.

Конфигурация процессора на основе потока данных представлена на рис. 1.11. Программа хранится в командных ячейках, содержащих код операции, несколько значений для входных операндов и список ячеек, которые должны получить результат, когда она будет выполнена. Во время работы процессора каждая ячейка ожидает, пока на место всех входных операндов поступят данные. Тогда она срабатывает и запрашивает в блоке арбитража обслуживание операционного устройства.

Блок арбитража рассматривает одновременно много сработавших командных ячеек и копии их содержимого направляет к свободным операционным устройствам. Эти устройства, организованные как конвейеры, выполняют операции. Результаты принимает блок распределения и направляет копии результатов в отведенные места тех командных ячеек, которые определены в списках пунктов назначения. Затем возможно срабатывание других ячеек и т. д.

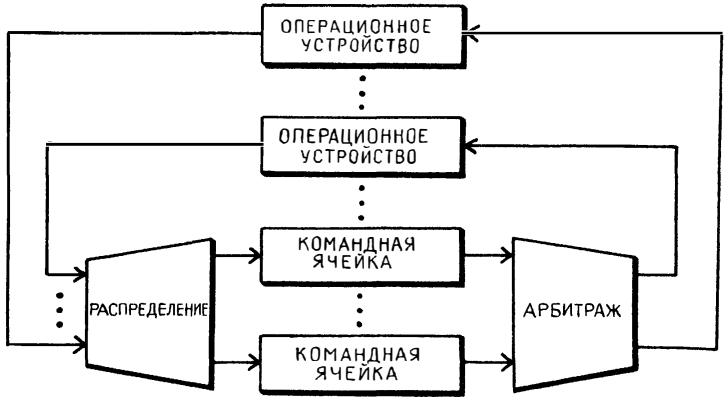

Рис. 1.11

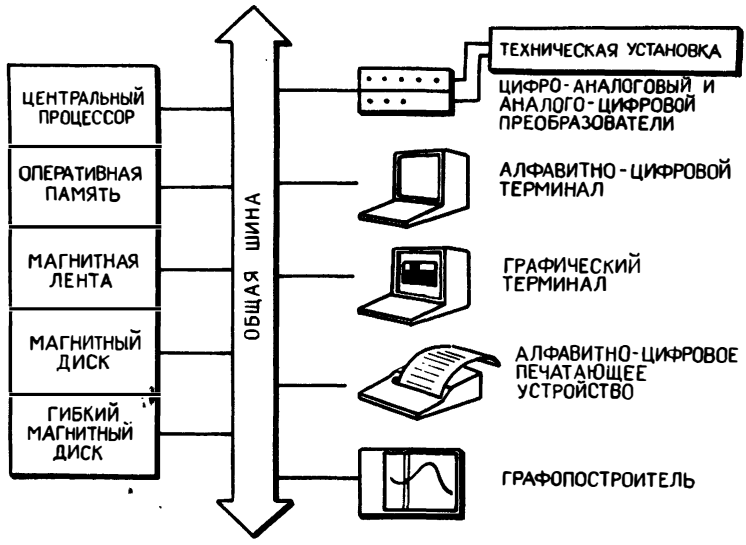

Рис. 1.12

Более подробно с вычислительными системами, содержащими новые архитектурные решения, можно познакомиться в [28].

● 1.4. Вычислительные системы серии СМ

1.4.1. Введение. Семейство мини-ЭВМ, называемое серией СМ, представляет собой ряд вычислительных машин, совместимых друг с другом снизу вверх. Совместимость снизу вверх означает, что программы, разработанные для младших моделей серии, будут выполняться на старших без изменений. Аналогом соответствующего ряда за рубежом является серия ЭВМ фирмы «DEC» (Digital Equipment Corporation), содержащая ЭВМ типа PDP.

Выбор ЭВМ серии СМ в качестве реальной вычислительной системы для иллюстрации архитектуры ЭВМ объясняется широкой популярностью этой серии в решении учебных, научно-технических, информационных задач среднего класса сложности, которым они соответствуют по своим техническим параметрам. Большое разнообразие периферийных устройств, подключаемых к ЭВМ, разнообразие операционных систем (см. гл. 4), применяемых в этой серии, выгодно выделяют машины СМ среди других ЭВМ.

Кроме того, отличительными особенностями серии мини-ЭВМ типа СМ (СМ 3, СМ 1300, «Электроника 100/16», СМ 4, СМ 1400, СМ 4/20, СМ 1420, СМ 1600, «Электроника 100/25», «Электроника 79») являются относительно невысокая стоимость, малая площадь размещения, легкое сопряжение с техническими установками. Штат, обслуживающий ЭВМ, может быть минимальным и состоять из инженера-электронщика и программиста, которые работают по необходимости: профилактика, генерация системы под новые тре-

бования пользователя и т. п.; остальное время на ЭВМ пользователь может работать самостоятельно.

Набор аппаратуры, который может быть включен в состав вычислительной системы (далеко не полный), показан на рис. 1.12. Центральный процессор, оперативная память и все внешние устройства подключаются к единой магистрали связи, называемой *общей шиной.*

Магистраль связи — общая шина — представляет собой набор проводов, передающих информацию, например данные, управляющие сигналы между устройствами, которые к ней подключены. Любое устройство, в том числе и центральный процессор, может поместить информацию в магистраль, и каждое устройство может выбрать ее из магистрали. Таким образом, общая шина является средством обмена информацией между устройствами. К такой системе легко подключаются новые устройства (дополнительный блок памяти или терминал).

Устройства могут обмениваться данными независимо от центрального процессора, что позволяет совмещать в одно и то же время выполнение нескольких работ. Дисплей может получить, например, информацию на экран из памяти, в то же время центральный процессор выполняет другую работу. Многие устройства, подключенные к общей шине, являются одновременно вводными и выводными (диски, терминалы).

1.4.2. Центральный процессор. Блок-схема центрального процессора приведена на рис. 1.13. Арифметическое и логическое устройство принимает информацию из общей шины, восьми общих регистров. Обработав информацию, устройство может поместить информацию в те же источники и в регистр состояния процессора. Для арифметических операций используется двухадресный формат команд, поэтому нет необходимости в регистре-аккумуляторе.

Устройство управления, явно не обозначенное на рис. 1.13, состоит из регистра состояния, счетчика команд (регистр R7) и общей шины.

Регистр состояния процессора содержит информацию о результате последней выполненной операции. Он состоит из 16 однобито-

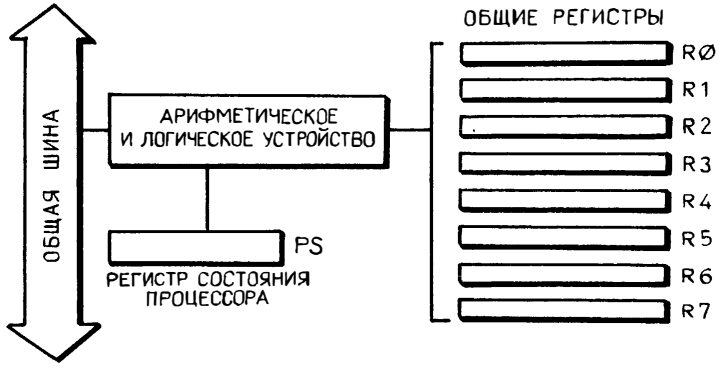

Рис. 1.13

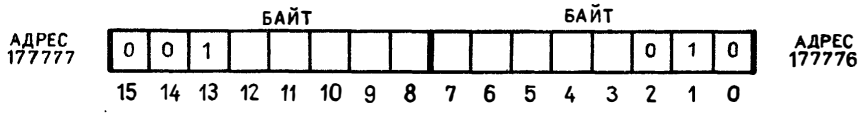

Рис. 1.14

вых индикаторов, и в зависимости от содержимого каждого бита можно определить, что произошло в результате операции (например, получен нулевой или отрицательный результат, имеет место переполнение и т. п.).

Счетчик команд (16-битовый регистр), как и в других ЭВМ, содержит адрес очередной команды, подлежащей выполнению. При обращении процессора к R7 для получения адреса ячейки (слова) оперативной памяти содержимое R7 увеличивается на 2, так как (это показано в следующем пункте) адрес слова всегда четный. Если появляется команда, требующая изменения последовательного хода выполнения программы, то в R7 устанавливается адрес передачи управления. Исходное содержание счетчика команд — стартовый адрес программы в машинных кодах.

1.4.3. Оперативная память. Память организована в виде последовательности слов-ячеек, каждое из которых содержит 16 бит. Состояние слова изображается диаграммой, приведенной на рис. 1.14. Все слова памяти пронумерованы. Номер слова называется его адресом.

Таким образом, с каждым элементом памяти связаны адрес элемента и его содержимое — информация в виде комбинации битов.

Отличие ЭВМ типа СМ от других машин состоит в том, что не только слово имеет собственный адрес, но его имеет и полуслово — байт, адрес младшего байта-биты (0—7) — всегда четный, совпадает с адресом слова.

Максимальный объем адресуемой памяти составляет 64 К байт или 32 К слов, где К = 1024, и может быть расширен специальным устройством — диспетчером памяти — до 4 М байт на старших моделях СМ ($М = К^2$).

Верхние (со старшими адресами) 4 К слов называются страницей ввода — вывода. Каждая ячейка в странице ввода — вывода является регистром управления или регистром данных, связанных с внешним устройством.

Некоторые ячейки резервируются системой и не используются программами пользователя. Обычно они имеют адреса 0—776. Верхние 4 К слов имеют адреса 160 000—177 776.

Заметим, что адреса памяти записываются в восьмеричной системе счисления, а числа (содержимое ячеек) — в двоичной системе.

1.4.4. Десятичная, восьмеричная, двоичная системы счисления. Все рассматриваемые здесь системы счисления относятся к так

называемым позиционным системам, в которых любое вещественное число представляется в виде

$$x = \pm a_n a_{n-1} \ldots a_0, a_{-1}, a_{-2} \ldots,$$

где значение x находится по формуле

$$x = \pm(a_n q^n + a_{n-1} q^{n-1} + \ldots + a_0 + a_{-1} q^{-1} + a_{-2} q^{-2} + \ldots).$$

Здесь целое число $q > 1$ — основание системы счисления, a_i — цифры q-ичной системы счисления.

Обычно в качестве младших цифр используют знаки 0 и 1. Основание системы записывают в виде 10. В десятичной системе: цифры 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, основание 10, пример записи числа:

$$103{,}12 = 1 \cdot 10^2 + 0 \cdot 10^1 + 3 \cdot 10^0 + 1 \cdot 10^{-1} + 2 \cdot 10^{-2}.$$

В восьмеричной системе: цифры 0, 1, 2, 3, 4, 5, 6, 7, основание 8 записывается в виде 10, пример записи числа:

$$103{,}12 = 1 \cdot 8^2 + 0 \cdot 8^1 + 3 \cdot 8^0 + 1 \cdot 8^{-1} + 2 \cdot 8^{-2}.$$

В двоичной системе: цифры 0, 1, основание 2 записывается в виде 10, пример записи числа:

$$101{,}11 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2}.$$

Приведем примеры записи целых чисел (в скобках отмечено основание системы счисления):

$$3_{(10)} = 3_{(8)} = 11_{(2)}; \quad 10_{(10)} = 12_{(8)} = 1010_{(2)}.$$

Использование двоичной системы, адекватной состояниям элементов ЭВМ, оказывается неудобным, так как трудно воспринимать двоичные числа из-за их громоздкой и непривычной записи. Перевод числа из десятичной системы в двоичную и наоборот выполняет машина. Однако, чтобы эффективно использовать ЭВМ, следует научиться интерпретировать слово машины. Для этого и используется восьмеричная система, числа которой читаются так же легко, как десятичные, и очень прост переход от двоичной системы к восьмеричной и наоборот.

Для перевода восьмеричного числа в двоичную систему необходимо каждую восьмеричную цифру заменить равным ей трехзначным двоичным числом, например

$$70{,}261_{(8)} = 11000{,}010110001_{(2)}; \quad 26{,}03_{(8)} = 010110{,}000011_{(2)}.$$

Для перевода двоичных чисел в восьмеричные необходимо начиная от запятой влево и вправо разбить двоичные цифры на тройки цифр, каждое трехзначное число перевести в восьмеричную систему, неполные тройки дополняют нулями, например

$$11{,}011111_{(2)} = 3{,}37_{(8)}; \quad 1011100{,}0111_{(2)} = 134{,}34_{(8)}.$$

1.4.5. Представление чисел в памяти. Целые числа занимают в памяти ЭВМ байт или слово (2 байта). Возможный диапазон

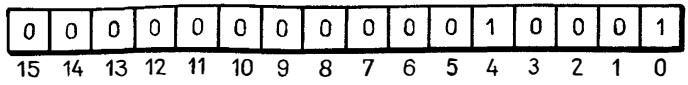

Рис. 1.15

изменения целых чисел без знака в формате слова 0—65 535, а со знаком — 32 768—32 767. Например, число 17 изобразится диаграммой, приведенной на рис. 1.15, при записи в формате слова. Вещественные числа имеют также два представления: одинарной и двойной точности. Вещественные числа одинарной точности занимают два слова, двойной — четыре слова памяти. Любое вещественное число x можно записать следующим образом:

$$x = m10^p,$$

где $|m|<1$, p — целое, $10=10_{(2)}$. Множитель m называется мантиссой, p — порядком числа x. Запись числа x называется нормализованной, если

$$\frac{1}{2}<|m|<1.$$

При записи числа в ненормализованном виде количество значащих цифр, изображающих число, может быть значительно меньше, чем при записи в нормализованном, и погрешность в представлении числа может быть больше.

Нормализованное число в одинарной точности записывается в память следующим образом (рис. 1.16): знак числа — в 15-м бите первого слова (0 — число положительное, 1 — число отрицательное), порядок p размещается в битах 7—14 первого слова, увеличенный на 128, мантисса занимает 23 бита в двух словах. Нормализованное число в двойной точности записывается в четыре слова памяти и отличается от одинарной точности только тем, что продолжение мантиссы размещается в $n+2$, $n+4$, $n+6$ словах или 55 битах.

Диапазон представления вещественных чисел $2{,}939\cdot10^{-39}_{(10)} \div 1{,}701\cdot10^{38}_{(10)}$. Точность представления числа в одинарной точности соответствует примерно 7 знакам после запятой в десятичном представлении числа (0,1234567), в двойной точности — 17 знакам (0,12345678901234567).

1.4.6. Представление букв и символов в памяти. Каждой букве и символу клавиатуры терминала соответствует число. Кодировка удовлетворяет стандарту КОИ-7, который, за исключением букв русского алфавита, совпадает с ASCII (American Standard Code for Information Interchange, произносится «эз-ки»). Символы, буквы и цифры терминала интерпретируются как восьмеричные числа от 0 до $177_{(8)}$, например

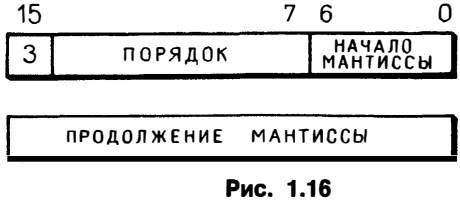

Рис. 1.16

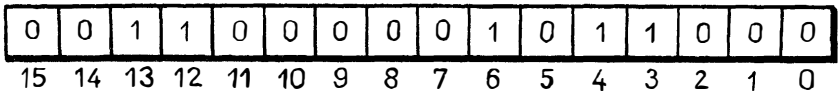

Рис. 1.17

A — 101	D — 104	X — 130	0 — 60
B — 102	E — 105	Y — 131	1 — 61
C — 103	F — 106	Z — 132	2 — 62

Для хранения символов терминала используется один байт памяти. Например, символы X0 в памяти запишутся в два байта (рис. 1.17).

● 1.5. Память на магнитных дисках и лентах

1.5.1. Общая схема магнитных дисков. Устройство внешней памяти на магнитных дисках состоит из следующих основных элементов (рис. 1.18): *1* — магнитный диск — носитель информации — круглая пластинка, покрытая магнитным материалом. Диск устанавливается на шпиндель двигателя вращения *5*. Информация записывается магнитными головками *4* на обе стороны диска. Магнитные головки перемещаются в радиальном направлении двигателем *6*. Вся поверхность диска разбита на дорожки *2*, равноудаленные от оси диска, а каждая дорожка делится на определенное число секторов *3*. Каждый сектор содержит один блок информации объемом 512 байт. Часто сектор информации, равный одному блоку, используют как единицу объема информации. Для управления чтением-записью служат электронные схемы *7*, которые вместе с *5*, *6* называют дисководом.

Используются магнитные диски четырех типов: фиксированные, кассетные, пакетные, гибкие. В фиксированных диск не снимается с дисковода. Кассетный диск представляет собой одну пластину, закрепленную в кассете, этот диск снимается с дисковода. Пакетный диск содержит несколько пластин, закрепленных на одной оси в кассете. Гибкий диск представляет собой полимерную пленку, покрытую магнитным материалом, помещенную в картонный футляр.

1.5.2. Устройство внешней памяти на магнитных дисках типа СМ 5400. Устройство СМ 5400 в настоящее время является одним из основных устройств внешней памяти для ЭВМ СМ. Устройство включает два диска: фиксированный и съемный. Емкость каждого диска 2,4 М байт. Скорость обмена данными с ЭВМ составляет 300 К байт/с. К ЭВМ может быть подключено несколько устройств типа СМ 5400.

На передней панели устройства СМ 5400 размещены переключатели и сигнальные лампочки, с помощью которых можно включить устройство, получить информацию о его

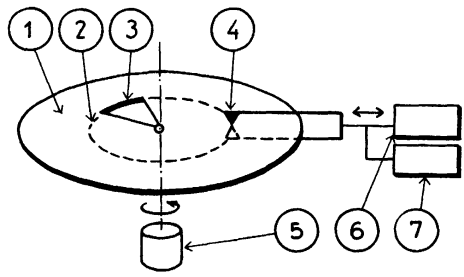

Рис. 1.18

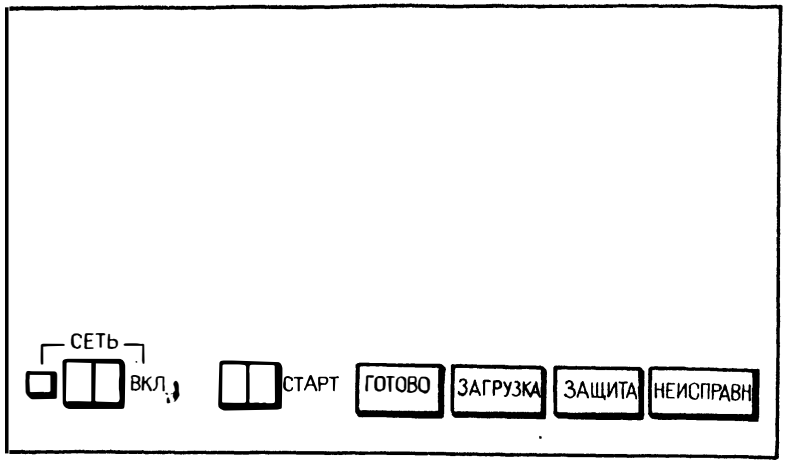

Рис. 1.19

работе (рис. 1.19). Подготовка устройства к работе состоит из следующих операций:

1) Вставить рабочий сменный диск.

2) Включить питание (нажать переключатель «Сеть»), загораются лампочки «Сеть» и «Загрузка».

3) Включить двигатель, вращающий диски. (Нажать переключатель «Старт»). Лампочка «Загрузка» гаснет. Подождать 20 с. Когда загорается лампочка «Готово», устройство готово к работе.

Если горит лампочки «Защита», необходимо отжать переключатель с надписью «Защита». Лампочка погаснет. Если горит лампочка «Неисправность», следует сообщить о неисправности устройства персоналу, обслуживающему ЭВМ. Нормальное состояние: горит только лампочка с надписью «Готово». Окончание работы с устройством состоит из следующих шагов.

1) Отключить двигатель, вращающий диски (отжать переключатель с надписью «Старт»). Лампочка «Готово» гаснет. Подождать, пока не загорится лампочка с надписью «Загрузка».

2) Отключить питание (отжать переключатель «Сеть»).

1.5.3. Устройство внешней памяти на гибких магнитных дисках типа СМ 5608. Устройство СМ 5608 может обслуживать два магнитных диска (им присваиваются номера 0 и 1 соответственно). Емкость одного магнитного диска примерно 256 К байт.

Переключатели на передней панели (рис. 1.20) обеспечивают: включение накопителя, замену дисков, выбор требуемой стороны данного диска. Подготовка устройства к работе:

1) Включить устройство.

2) Открыть защитную дверцу и вставить гибкий диск нужной стороной в соответствующее гнездо.

3) Установить переключатель сторон данного диска в положение, соответствующее стороне диска, с которой будет производиться работа. (Переключатель сторон А/В в нажатом состоянии опреде-

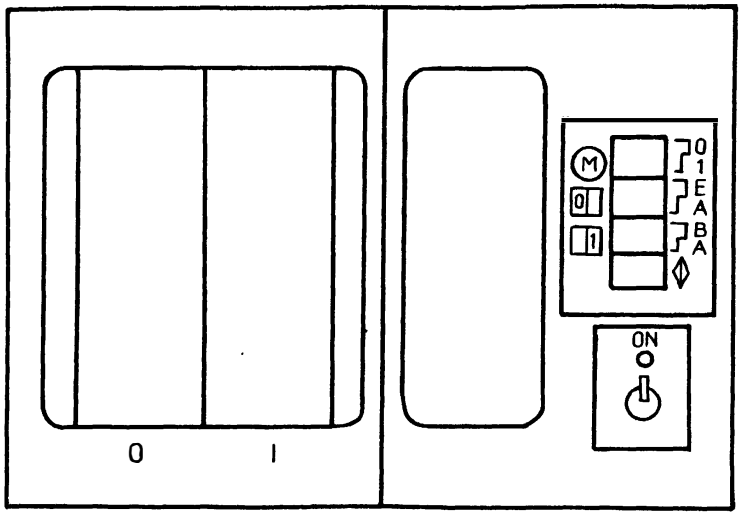

Рис. 1.20

ляет сторону А, в отжатом состоянии — сторону В). В настоящее время возможна работа только на стороне А гибкого диска, т. е. необходимо нажать переключатель сторон А/В для диска с нужным номером.

4) Включить двигатель, вращающий диски (нажать переключатель с надписью «М»).

Устройство к работе готово. Защитные дверцы блокируются (нельзя открыть).

Замена диска состоит из следующих операций:

1) Деблокировать защитные дверцы (нажать и отжать переключатель с надписью «М»). Подождать несколько секунд (до появления характерного щелчка). Двигатель, вращающий диски, отключается.

2) Открыть защитную дверцу и заменить диск.

3) Установить переключатель сторон данного диска в нужное положение.

4) Закрыть защитную дверцу. Двигатель, вращающий диски, включается. Устройство к работе готово.

Окончание работы:

1) Отключить двигатель, вращающий диски (отжать переключатель с надписью «М»).

2) Отключить питание.

1.5.4. Память на магнитных лентах. Магнитные ленты применяются для длительного хранения большого объема информации, переноса информации с одной ЭВМ на другую, хранения копий важной для пользователя информации с магнитных дисков.

Информация записывается на полимерную ленту, покрытую магнитным составом, шириной 12,5 мм. Лента наматывается на катушки, вмещающие 80, 375, 750 м. Запись идет по девяти дорожкам; на восьми записывается информация, девятая —

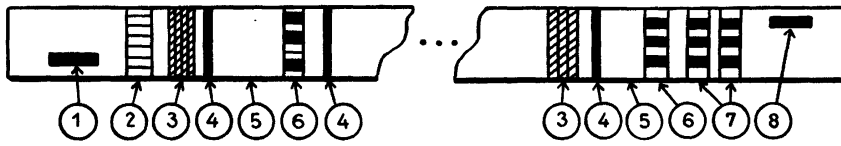

Рис. 1.21

контрольная. Плотность записи (количество битов на 1 мм по длине одной дорожки) — важный показатель магнитной ленты и устройства чтения-записи; возможные значения плотности: 8, 32, 64, 256 бит/мм.

Общая структура записи на ленту показана на рис. 1.21. Запись информации пользователя называется *файлом*, если ей присвоено имя. Структура информации: *1* — физическое начало ленты (светоотражающая полоска на нерабочей стороне ленты), *2* — заголовок ленты, *3* — заголовок файла, *4* — разделитель группы записей, *5* — записи файла, *6* — метка конца файла, *7* — две метки конца файла — конец ленты, *8* — физический конец ленты.

Рассмотрим устройство внешней памяти на магнитной ленте типа СМ 5300. Емкость одной кассеты около 10 М байт. Продольная плотность записи 32 бит/мм. Скорость обмена информацией с ЭВМ составляет 10 К байт/с. Для предотвращения случайной записи на магнитную ленту используется кольцо разрешения записи на кассете. При отсутствии кольца запись информации на магнитную ленту запрещена. На передней панели устройства размещены переключатели и сигнальные лампочки (рис. 1.22).

Подготовка устройства к работе состоит из следующих операций:

1) Заправить ленту в устройство.

2) Включить устройство (нажать переключатель с надписью «ВКЛ»).

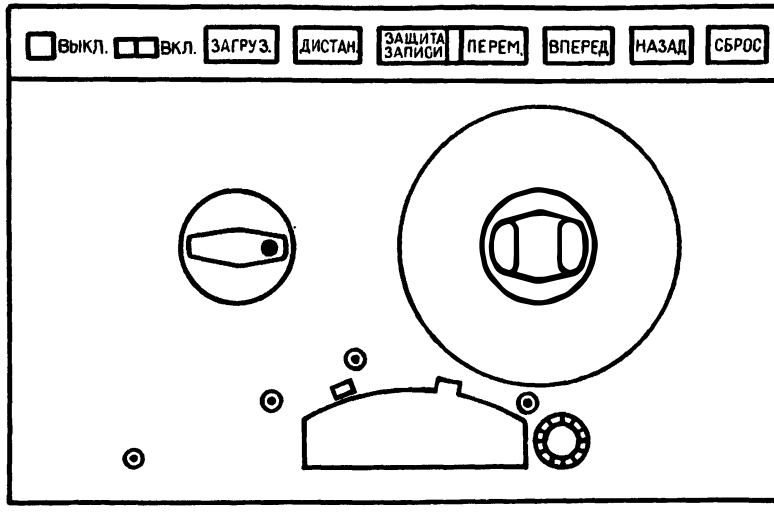

Рис. 1.22

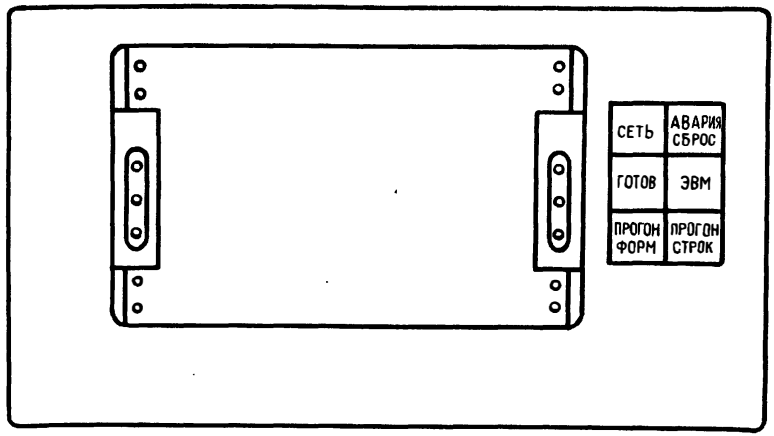

Рис. 1.23

3) Нажать переключатель «Загрузка». Магнитная лента устанавливается на маркере начала ленты. Загорается лампочка «Дистанц.». Устройство к работе готово.

Если отсутствует кольцо защиты записи, то также загорается лампочка «Защита записи».

После окончания работы лента должна быть перемотана на исходную катушку. Для этого необходимо нажать последовательно переключатели «Сброс», «Перемотка». Лента остановится на маркере начала. Повторно нажать переключатели «Сброс», «Перемотка». После перемотки магнитной ленты выключить устройство (отжать переключатель с надписью «ВКЛ»).

● 1.6. Алфавитно-цифровые печатающие устройства

Алфавитно-цифровые печатающие устройства (АЦПУ) служат для вывода информации на бумагу — документирования результатов работы пользователя. Рассмотрим два типа АЦПУ: параллельное и последовательное,— которые отличаются скоростью вывода информации.

1.6.1. АЦПУ параллельное типа СМ 6315. Устройство СМ 6315 предназначено для вывода информации на бумажный носитель. Скорость печатания 500 строк/мин. Ширина строки до 128 символов.

На передней панели устройства расположены переключатели и сигнальные лампочки (рис. 1.23).

Подготовка устройства к работе:

1) Включить устройство (нажать тумблер в нижней части устройства). Загораются лампочки «Сеть», «Авария». Через 40 с лампочка с надписью «Авария» гаснет. Загорается лампочка «Готов». Устройство находится в автономном режиме. В этом режиме нажатие переключателя «Прогон строк» вызывает перемещение бумаги на одну строку, а нажатие переключателя «Прогон формата» вызывает перемещение бумаги на одну страницу.

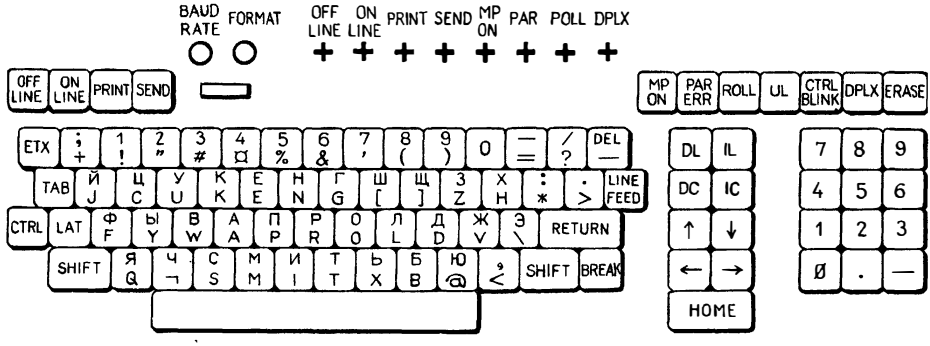

Рис. 1.24

В том случае, если лампочка «Авария» не гаснет, сообщить персоналу, обслуживающему ЭВМ.

2) Для подключения устройства к ЭВМ необходимо нажать переключатель с надписью «ЭВМ». Загорается соответствующая лампочка.

Окончание работы с АЦПУ:

1) Перевести устройство в автономный режим (нажать переключатель с надписью «ЭВМ»). Соответствующая лампочка гаснет.

2) Отключить устройство (отжать тумблер в нижней части устройства).

1.6.2. Последовательное печатающее устройство типа DZM-180. DZM-180 является печатающим устройством со скоростью печатания 180 зн/с. Относительно низкая стоимость, надежность работы обусловили ее широкое применение в качестве устройства вывода для ЭВМ СМ.

На передней панели устройства расположены переключатели и сигнальные лампочки (рис. 1.24). Подготовка устройства к работе:

1) Включить устройство (нажать переключатель с надписью «SUPPLY»). Загорается красная лампочка (это автономный режим работы).

2) Перевести устройство в режим работы с ЭВМ (нажать переключатель с надписью «SEL»). Красная лампочка гаснет. Загорается зеленая лампочка. Устройство к работе готово.

Окончание работы:

1) Перевести устройство в автономный режим (нажать переключатель с надписью «На»). Загорается красная лампочка.

2) Отключить устройство (нажать переключатель с надписью «SUPPLY»).

● 1.7. Алфавитно-цифровой терминал

Рассмотрим терминалы VDT («Видеотон») и «Электроника». Терминалы других типов могут иметь некоторые отличия (другое расположение клавиатуры, новые функциональные возможности).

При работе с ними необходимо предварительно ознакомиться с соответствующим руководством.

Алфавитно-цифровой терминал предназначен для ввода — вывода символьной информации. Экран терминала делится на 16 (или 24) строк по 80 символов в каждой строке.

Набор символов VDT (см. рис. 1.24):

26 заглавных латинских букв;

31 заглавная русская буква;

10 цифр;

28 специальных знаков.

После включения терминала в первой позиции первой строки появляется мерцающая метка (курсор). Для работы с терминалом необходимо знать назначение следующих клавиш.

Клавиша OFF LINE. После включения устройства устанавливается режим OFF LINE и на панели загорается индикаторная лампочка «OFF LINE». Если устройство работает, то при нажатии этой клавиши устройство переходит в режим OFF LINE. В данном режиме линия связи с ЭВМ отключена. Терминал «Электроника» не имеет такой клавиши.

Следующая клавиша ON LINE. Нажатие этой клавиши переводит терминал из режима OFF LINE в режим ON LINE. Терминал подключается в линии связи с ЭВМ. Загораются индикаторные лампочки «ON LINE» и «DPLX».

Рассмотрим действие клавиши SHIFT для терминала VDT и клавиш ВР, НР для «Электроники». При совместном нажатии SHIFT с алфавитно-цифровыми клавишами вырабатывается код верхнего знака, изображенного на выбранной клавише. В отжатом состоянии — код нижнего знака. При совместном нажатии алфавитно-цифровой клавиши и клавиши «ВР» вырабатывается код верхнего знака (в случае «НР» — код нижнего знака).

Клавиша «CTRL» для VDT или аналогичная клавиша «СУ» для «Электроники» служит для генерирования управляющих кодов. Подробности использования этой клавиши излагаются в гл. 4.

Клавиша «LINE FEED» для VDT или ПС для «Электроники». При нажатии этой клавиши курсор переводится в первую позицию следующей строки.

Клавиша «TAB» для VDT или аналогичная клавиша ГТ для «Электроники». При нажатии клавиши курсор устанавливается в ближайшую позицию для тубуляции. Обычно это позиции строки с номерами $8N+1$, где $N=0, 1, 2, \ldots$ (номера позиций задаются аппаратно).

Клавиша «ROLL» служит для установления и отмены режима «бегущий экран». При записи символа в последнюю позицию последней строки метка не переходит в первую позицию первой строки, как в случае обычного режима, а вместо этого все строки сдвигаются на одну строку вверх, причем первая строка теряется, а на месте последней строки образуется пустая строка. Курсор устанавливается в первую позицию последней строки.

Клавиша «RETURN» для VDT или ВК для «Электроники». Нажатие клавиши «RETURN» служит признаком конца строки символов, введенной с терминала (при работе в операционной системе ОС РВ).

При нажатии алфавитно-цифровых клавиш вырабатывается код, соответствующий данному символу.

Подготовка терминала к работе (VDT):

1) Включить терминал. Загорается лампочка «OFF LINE» (автономный режим работы).

2) Нажать клавишу «ON LINE». Загорается лампочка «ON LINE» и DPLX (режим связи с ЭВМ).

3) Нажать клавишу «ROLL». Загорается лампочка «ROLL». (Установить «Бегущий экран»).

Окончание работы (VDT):

1) Нажать клавишу «OFF LINE». Загорается соответствующая лампочка.

2) Выключить терминал.

Глава 2

СТРУКТУРА АЛГОРИТМОВ И ПРОГРАММ

● 2.1. Семейства вычислительных алгоритмов

Ниже перечислены и кратко описаны основные семейства алгоритмов, которые применяются в инженерно-технических задачах. Каждый алгоритм семейства рассматривается как один-единственный вычислительный блок. При таком подходе алгоритм есть отображение входных данных задачи на выходные данные. Как устроено это отображение, нас в данный момент не интересует, так же как и вопрос о его реализации на ЭВМ.

Можно представить, что имеется гипотетический калькулятор, содержащий клавиши и несколько кнопок, есть возможность ввести в него исходные данные задачи, нажать одну из клавиш и одну из кнопок и вывести ответ — выходные данные (рис. 2.1). Собственно, так и поступают, работая с калькулятором. Мы не задумываемся, что, набрав на индикаторе число 0,37, нажав клавишу «F» и кнопку «SIN», мы заставляем калькулятор выполнять цепочку действий; нас интересует только ответ 0,36161543 на индикаторе. Каждая из упомянутых клавиш отвечает какому-либо семейству алгоритмов, а кнопка — конкретному алгоритму из этого семейства.

Семейство алгоритмов будем обозначать буквой латинского алфавита и десятичной цифрой, например A3, B7, а конкретный алгоритм отмечать парой буква — цифра. Таким образом, обозначение A5B2 означает: алгоритм B2 из семейства A5.

Ниже, если это специально не оговаривается, входными и выходными данными алгоритма могут быть как точные, так и приближенные значения решения математической задачи. Поэтому смысл, придаваемый выражениям «решение уравнения», «решение системы» и т. п., должен устанавливаться при описании конкретного алгоритма семейства. Рассмотрим следующие семейства.

A0. Вычисление элементарных функций. Обычный набор элементарных функций: $\sin x$, $\cos x$, $\mathrm{tg}x$, $\arcsin x$, $\arccos x$, $\mathrm{arctg}x$, e^x, $\sqrt{x}$, $\ln x$, $\lg x$, $|x|$, $\mathrm{th}x$ дополняется часто встречающимися $y=\max(x_1, x_2, \ldots, x_n)$, $y=\min(x_1, x_2, \ldots, x_n)$ и некоторыми другими (см. гл. 12).

Рис. 2.1

На вход любого блока A0 следует подать

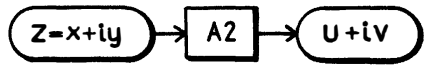

Рис. 2.2

числа из области определения функции (например, $x \geqslant 0$ для $\sqrt{x}$, $\ln x$, $\lg x$), на выходе получается число из области значений функции.

А1. Вычисление специальных функций. Блок семейства А1 представляет собой полный аналог А0, и только исторически более позднее исследование этих функций закрепило за ними термин «специальные». К числу специальных функций относятся: интегральный синус $\mathrm{Si}(x)$, интегральный косинус $\mathrm{Ci}(x)$, гамма-функция $\Gamma(x)$, функции Бесселя $J_0(x)$, $J_1(x)$, $Y_0(x)$, $Y_1(x)$, модифицированные функции Бесселя $K_0(x)$, $K_1(x)$, $I_0(x)$, $I_1(x)$, интегралы Френеля $S(x)$, $C(x)$, эллиптические интегралы, интеграл вероятности и др.

А2. Вычисление элементарных функций комплексного переменного. Алгоритмы семейства А2 для своей работы должны иметь на входе комплексное число z. На выходе алгоритм выдает комплексное число $f(z)=u+iv$ (рис. 2.2), при $v=0$ — вещественное число. Часто используются следующие функции: $|z|$, $\ln z$, $\sqrt{z}$, $\sin z$, $\cos z$ и т. п.

А3. Аппроксимация функций. Задача приближения заданной функции $f(x)$ другой функцией $\varphi(x)$ из некоторого класса функций решается алгоритмами семейства А3. Входными данными для А3 являются требуемая точность приближения ε и $f(x)$, заданная либо аналитической формулой, например, $f(x)=x^2+\sin(\operatorname{arctg} x+\cos x)$, $0 \leqslant x \leqslant 1$, либо таблицей значений

x_j	0,0	0,2	0,3	0,4	0,6	0,9	1,0
$f(x_j)$	1,1	2,0	1,9	0,7	1,4	1,6	0,9

Выходные данные — достигнутая точность приближения ε_1 и функция $\varphi(x)$, $0 \leqslant x \leqslant 1$, которая также может быть представлена в аналитическом или в табличном виде (рис. 2.3).

А4. Интегрирование. В семействе А4 алгоритмы вычисляют определенные интегралы от функций $f(x)$ одной или нескольких переменных. На вход алгоритмов подают точность ε, пределы интегрирования, функцию $f(x)$ в аналитическом или табличном виде. На выходе получаем число S — приближенное значение интеграла $S=\int_a^b f(x)\,dx$. Алгоритмы А4, рассматриваемые в книге, решают задачу интегрирования методами численного анализа. Однако существуют на ЭВМ системы аналитических вычислений, которые имеют алгоритмы нахождения аналитических формул для неопределенного интеграла. На входе этих алгоритмов — формула для $f(x)$, на выходе — формула для $\int f(x)dx$. Поэтому блок А4 следует рассматривать как блок, вычисляющий определенные и неопределенные интегралы.

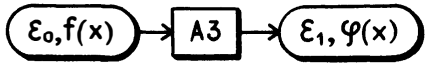

Рис. 2.3

А5. Суммирование рядов. Алгоритмы А5 на входе имеют формулы для общего члена ряда: числового a_n, функционального $a_n(x)$ и точность ε суммирования, на выходе получаем приближенное значение $S=\sum_{n=1}^{\infty} a_n$ или $S(x)=\sum_{n=1}^{\infty} a_n(x)$, $S(x)$ предствляется аналитически или таблично.

А6. Фурье-анализ. Разложение функции $f(x)$ в ряд Фурье в вещественной форме

$$f(x)\sim\frac{1}{2}a_0+\sum_{n=1}^{\infty}(a_n\cos nx+b_n\sin nx),\quad \pi\leqslant x\leqslant\pi,$$

$$a_n=\frac{1}{\pi}\int_{-\pi}^{\pi}f(x)\cos nx\,dx,\quad b_n=\frac{1}{\pi}\int_{-\pi}^{\pi}f(x)\sin nx\,dx,$$

или в комплексной форме

$$f(x)\sim\sum_{n=-\infty}^{+\infty}c_n e^{inx},\quad c_n=\frac{1}{\pi}\int_{-\pi}^{\pi}f(x)e^{-inx}dx$$

составляет предмет Фурье-анализа. Фурье-анализ лежит в основе методов решения многих инженерно-технических задач, таких, как выявление скрытых периодичностей в динамических процессах, решение линейных уравнений математической физики методом Фурье и т. д. На входе алгоритмов А6 имеем $f(x)$ в аналитической или табличной (часто называемой дискретной) форме, на выходе — коэффициенты ряда Фурье. Заметим, что коэффициенты ряда Фурье — это функция $C=C(n)$ от целочисленного аргумента. Если n заменить вещественным ω, то преобразование функции $f(x)$ в $C(\omega)$ называют Фурье-преобразованием. Алгоритмы А6 выполняют также Фурье-преобразование.

А7. Дифференцирование. Вычислительный блок дифференцирования функции одной $f(x)$ или нескольких переменных $f(x)=f(x_1, x_2, ..., x_n)$ имеет на входе аналитическую или табличную форму $f(x)$, на выходе — аналитическую или табличную форму производной $f^{(p)}_{x^p}(x)$. Аналитическое дифференцирование представлено в системах аналитических вычислений на ЭВМ. В этом случае блок А7 дает точную аналитическую формулу для $f^{(p)}_{x^p}(x)$. Алгоритмы численного дифференцирования А7 на выходе имеют приближенную функцию $\varphi(x)$ к $f^{(p)}_{x^p}(x)$, $\varphi(x)$ может иметь как аналитическую, так и табличную формы.

А8. Операции с матрицами и векторами. Алгоритмы этого семейства реализуют основные операции линейной алгебры. На входе алгоритмов задаются матрицы и векторы, на выходе имеем результат операции. Примерами таких операций являются сумма, произведение матриц, обращение матрицы, скалярное произведение векторов, ортогонализация векторов и т. п.

А9. Решение систем линейных уравнений. На входе алгоритмов семейства А9 задаются матрица системы линейных уравнений

A и вектор правых частей b системы уравнений $Ax=b$; на выходе для матрицы с $\det A \neq 0$ получаем вектор x — решение системы.

B0. Собственные значения и векторы. На входе алгоритмов B0 задается матрица A, на выходе выдается система собственных векторов e_i, $i=1, 2, ..., m$, и соответствующих собственных значений λ_i, удовлетворяющих уравнениям

$$Ae_i=\lambda_i e_i, \quad 1 \leqslant i \leqslant m.$$

B1. Линейная оптимизация. Входом канонической задачи линейной оптимизации является вектор $c=(c_1, c_2, ..., c_n)$, определяющий скалярное произведение $\Phi(x)$ с неизвестным вектором x, у которого координаты $x_i \geqslant 0$, $1 \leqslant i \leqslant n$,

$$\Phi(x)=(c, x)=c_1x_1+...+c_nx_n.$$

Матрица A имеет размер $m \times n$, вектор $b=(b_1, ..., b_m)$. Они описывают ограничения типа равенств

$$Ax=b.$$

На выходе алгоритмов B1 имеем вектор x, доставляющий $\min \Phi(x)$, либо сообщение о неразрешимости задачи.

B2. Решение линейных интегральных уравнений. Интегральное уравнение общего типа можно записать следующим образом:

$$Ax-\lambda x=b.$$

Входом алгоритмов семейства B2 являются функции $A(t, s)$ — ядро интегрального оператора A

$$Ax(t)=\int_{D_0} A(t, s)x(s)ds, \quad t \in D_1,$$

область интегрирования D_0, область значений D_1, число λ, функция $b(t)$. Выходом B2 является функция $x(t)$ — решение интегрального уравнения — либо информация о неразрешимости или бесчисленном множестве решений.

B3. Корни полиномов. Алгоритмы семейства B3 по входным данным — коэффициентам a_i, $0 \leqslant i \leqslant n$, — определяют корни x_i, $1 \leqslant i \leqslant n$, полинома

$$P_n(x)=a_0+a_1x+a_2x^2+...+a_nx^n.$$

Следует заметить, что до 4-й степени $(n \leqslant 4)$ корни $P_n(x)$ могут вычисляться по коэффициентам аналитически.

B4. Корни нелинейных уравнений. Алгоритмы решения нелинейных уравнений B4 на входе имеют нелинейную функцию $f(x)$ одной переменной или систему нелинейных функций нескольких переменных $f(x)=f_i(x_1x_2, ..., x_n)$, $1 \leqslant i \leqslant n$, в области D, точность определения корня, на выходе алгоритмы B4 выдают корни (один или несколько) уравнений $f(x)=0$ в области D.

B5. Нелинейная оптимизация. Входные данные алгоритмов B5 — нелинейная функция $\Phi(x)$, $x=(x_1, ..., x_n)$ и область D переменных

x, где ищется минимум или максимум $\Phi(x)$. Область D может быть задана, например, неравенствами $\varphi(x) \geqslant 0$, $1 \leqslant i \leqslant m$, и равенствами $\psi(x)=0$, $1 \leqslant j \leqslant k$. На выходе алгоритмов — вектор x, доставляющий $\min \Phi(x)$ или $\max \Phi(x)$ в области D либо сообщение о неразрешимости задачи.

В6. Решение обыкновенных дифференциальных уравнений, задача Коши. На входе алгоритмов В6 задаются вектор-функция $f_i(x, y_1, ..., y_n)$, $1 \leqslant i \leqslant n$, интервал $a \leqslant x \leqslant b$ двумя числами a, b и начальный вектор $y^{(0)}=(y_1^0, y_2^0, ..., y_n^0)$, точность ε. На выходе — вектор-функция $y(x)=(y_1(x), y_2(x), ..., y_n(x))$, $a \leqslant x \leqslant b$,— решение задачи

$$\frac{dy_i}{dx}=f_i(x, y_1, ..., y_n), \quad y_i(a)=y_i^0 .$$

В7. Решение обыкновенных дифференциальных уравнений, краевые задачи. Входом алгоритмов В7 являются вектор-функция $f_i(x, y_1, ..., y_n)$, $1 \leqslant i \leqslant n$, интервал $a \leqslant x \leqslant b$, заданный числами a, b, вектор-функция $\varphi_i(y_1, ..., y_n)$, $1 \leqslant i \leqslant n$. На выходе — $y(x)=(y_1(x), ..., y_n(x))$ — решение краевой задачи

$$\frac{dy_i}{dx}=f_i(x, y_1, ..., y_n),$$

$$\varphi_s(y_1(a), ..., y_n(a))=0, \quad 1 \leqslant s \leqslant n,$$

$$\varphi_k(y_1(b), ..., y_n(b))=0, \quad 1 \leqslant k \leqslant n,$$

или сообщение о неразрешимости или бесчисленном множестве решений.

В8. Решение уравнений с частными производными. На входе алгоритмов В8, как и В7, задаются функции, описывающие дифференциальное уравнение, например, вида

$$\frac{\partial u}{\partial t}=f\left(u, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}\right).$$

Здесь f — функция на входе В8. Кроме того, на входе задается область D изменения независимых переменных. Для рассматриваемого уравнения область D может быть следующая:

$$D=\{(x, t): t_0 \leqslant t \leqslant t_1, \ x_0 \leqslant x \leqslant x_1\},$$

где x_i, t_i — заданные числа. Входными данными алгоритмов В8 являются дополнительные условия, однозначно определяющие решение $u(x, t)$ уравнения. Обычно это начальные и граничные условия, например

$$u(x, t_0)=\varphi(x), \quad x_0 \leqslant x \leqslant x_1,$$

$$u(x_0, t)=\mu_0(t); \ u(x_1, t)=\mu_1(t), \ t_0 \leqslant t \leqslant t_1,$$

где $\varphi(x)$, $\mu_0(t)$, $\mu_1(t)$ — заданные функции на входе алгоритма. Выходными данными является решение уравнения $u(x, t)$.

● 2.2. Структура алгоритмов

В гл. 1 определено понятие архитектуры ЭВМ как набора ресурсов ЭВМ, их структура, доступные программисту. Программист-пользователь может вообще не знать, как физически устроена и работает ЭВМ. Это знает инженер-электронщик. Для программирования важна логическая организация машины. По аналогии с архитектурой ЭВМ можно ввести понятие архитектуры алгоритма.

Архитектура алгоритма — это набор элементарных вычислительных блоков, их структура и логическая организация, необходимые для программирования задачи.

Программист-пользователь может не знать, как запрограммирован отдельный вычислительный блок и даже какой метод вычислений заложен в его основу, это должен знать инженер-математик. Хотя и не все инженеры-математики могут ответить на вопрос, по какому алгоритму вычисляются значения элементарных функций.

Естественно задать вопрос: что же должен уметь пользователь при построении алгоритма решения задачи и программировании?

Пользователю необходимо уметь:

1) Спроектировать алгоритм методом «сверху вниз» — от сложного к простому, используя готовые (базовые) вычислительные алгоритмы из семейств, перечисленных в 1.1, и применяя «базовые логические схемы».

2) Выбрать те вычислительные алгоритмы, которые удовлетворяют требованиям задачи.

3) В случае отсутствия готового вычислительного алгоритма необходимо его разработать, запрограммировать и включить в состав базовых вычислительных алгоритмов.

4) Наконец, необходимо запрограммировать весь алгоритм.

2.2.1. Проектирование алгоритма. Проектирование алгоритма удобно сопровождать блок-схемами. В блок-схемах приняты стандартные графические средства, показанные на рис. 2.4. Внутрь графического знака записывается кратко действие алгоритма, а в знак «связь» записывается метка, которая обозначает соединение с другой линией, имеющей кружок с такой же меткой. В ромб записываются анализируемые условия, а возможные результаты анализа обозначаются на выходящих из ромба линиях.

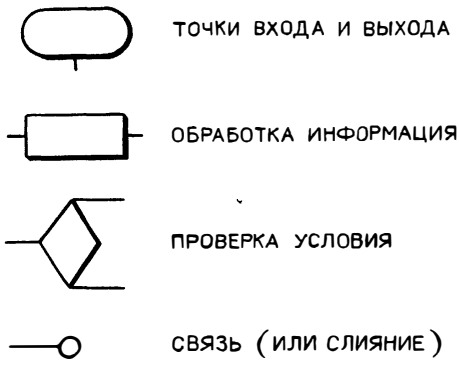

Рис. 2.4

Имеются три *базовые логические схемы*, на основании которых можно спроектировать любой вычислительный алгоритм.

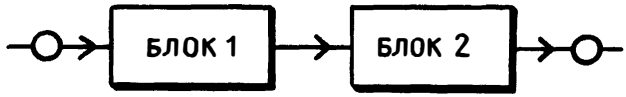

Рис. 2.5

Последовательная схема (рис. 2.5). Состоит из трех компонентов: входа в вычислительный блок *1*, вычислительного блока *2* и выхода из вычислительного блока *2*.

Альтернативная схема (рис. 2.6). Состоит из проверки выполнения некоторого условия. Если условие выполняется, то вычисления производятся на блоке *1*, если — нет, то на блоке *2*. Заметим, что в схеме один из блоков может быть пустым (не содержит действий).

Схема повторения или схема цикла. Состоит из узла слияния, проверки условия и вычислительного блока (рис. 2.7). В этой схеме после узла слияния происходит проверка условия: если оно выполняется, то происходят вычисления на блоке,

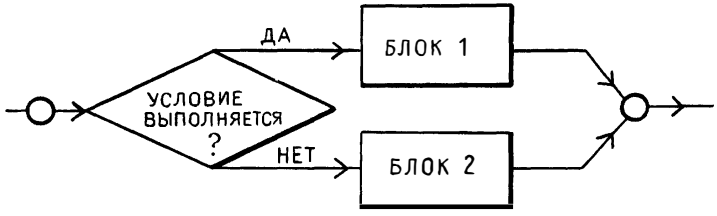

Рис. 2.6

если — нет, то происходит выход из схемы. Таким образом, вычисления в блоке выполняются до тех пор, пока условия не будут выполняться. Поскольку условие проверяется до вычислительного блока, возможно, что действие блока не выполнится ни разу. С другой стороны, если условие всегда выполняется, то вычислительный процесс зацикливается, так как нет возможности попасть на выход схемы.

В качестве примера приведем блок-схему алгоритма суммирования кубов целых чисел (рис. 2.8)

$$S=\sum_{k=1}^{N} k^3.$$

Число N — входное данное, S — выходное.

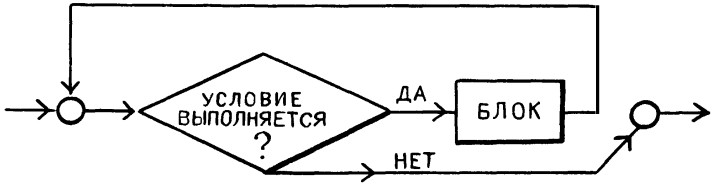

Рис. 2.7

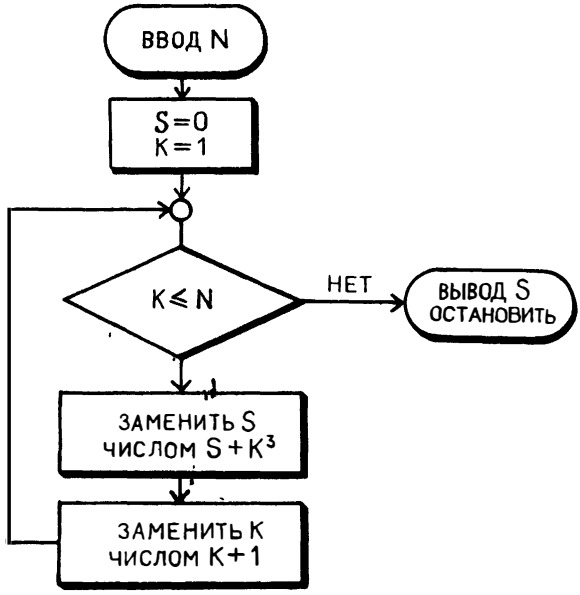

Рис. 2.8

Проектирование алгоритма методом «сверху вниз» на основе базовых логических схем и базовых алгоритмов состоит из следующих шагов. Исходный вычислительный алгоритм представляется в виде одного блока нулевого уровня, на вход которого подаются исходные данные задачи, на выходе получаются выходные данные. *Базовыми алгоритмами* будем называть те из алгоритмов описанных выше семейств, которые могут быть реализованы пользователем на его вычислительной машине. Например, совокупностью базовых алгоритмов можно считать алгоритмы, для которых представлены в гл. 12 описания фортран-программ.

Если задача может быть решена с помощью одного базового алгоритма, то проектирование на этом заканчивается, если — нет, то нулевой уровень проекта алгоритма разворачивается в последовательность базовых алгоритмов (рис. 2.9), где вычислительные блоки можно обозначить по принятой схеме: A4A0, B1A1 и т. п. Не помеченные на рис. 2.9 вычислительные блоки отсутствуют среди базовых алгоритмов и должны быть разработаны пользователем, обозначены в соответствии с принятой схемой и включены в состав базовых.

Таким образом, проект алгоритма первого уровня состоит в последовательном обращении к базовым алгоритмам и не содержит 2-й и 3-й логических схем в своей структуре.

Начиная со второго уровня проекта алгоритма используются все три логические схемы и базовые алгоритмы. Второй уровень представляет собой проекты отсутствующих алгоритмов на первом уровне (рис. 2.10). Непомеченные блоки второго уровня разрабатываются на следующем, третьем, уровне проекта и т. д. до тех пор, пока на n-м уровне проекта все вычислительные блоки не будут состоять из базовых алгоритмов. Алгоритмы вычислительной

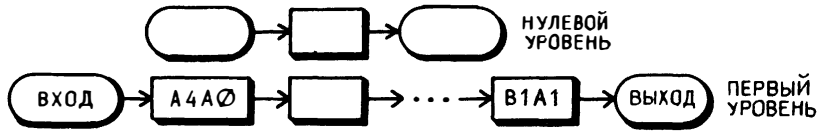

Рис. 2.9

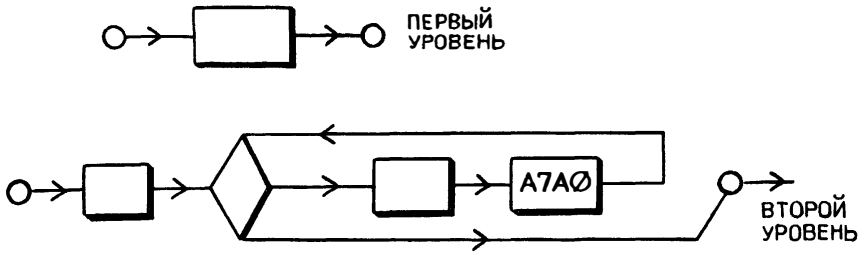

Рис. 2.10

математики, встречающиеся в инженерно-технических задачах, проектируются с помощью базовых алгоритмов с небольшим числом уровней проекта. Конечность этой процедуры (n конечно) следует из условия, что в число базовых алгоритмов естественно включаются арифметические действия над числами (рис. 2.11).

Отличительной чертой нисходящего проектирования алгоритма является возможность контроля его работы с самого начала

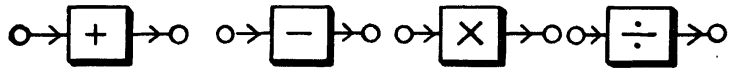

Рис. 2.11

проектирования. Это достигается тем, что неразработанные вычислительные блоки заменяются так называемыми «заглушками». Заглушки — это блоки, имитирующие вход и выход неразработанных блоков (рис. 2.12). Заглушка нужна только для того, чтобы проверить алгоритм определенного уровня.

2.2.2. Пример проектирования алгоритма. Пусть требуется спроектировать алгоритм решения следующей технической задачи. По измеренному переходному процессу $y(t)$ системы автоматического регулирования необходимо найти коэффициент k одного из ее элементов, который лежит в некотором диапазоне $k_0 \leqslant k \leqslant k_1$. Переходный процесс может быть с требуемой точностью описан решением системы обыкновенных дифференциальных уравнений

$$\frac{dx_1}{dt} = t + x_1 x_2, \quad x_1(0) = x_1^0, \quad 0 \leqslant t \leqslant T,$$

$$\frac{dx_2}{dt} = x_1 - k x_2, \quad x_2(0) = x_2^0. \tag{2.2.1}$$

Предположим, что математический метод решения этой задачи состоит в отыскании такого значения k, которое минимизирует норму S отклонения теоретического переходного процесса $x(t)$ от измеренного $y(t)$:

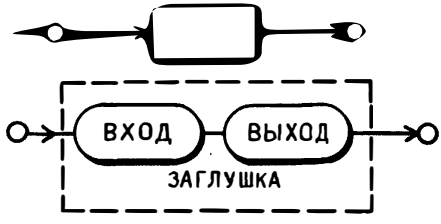

Рис. 2.12

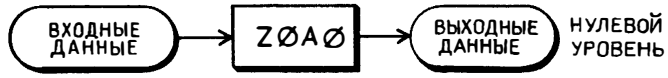

Рис. 2.13

$$S=\max_{0\leqslant t\leqslant T}\ \max_{1\leqslant i\leqslant 2}|x_i(t)-y_i(t)|.$$

Значение k, дающее $\min\limits_{k_0\leqslant k\leqslant k_1} S$, принимается в качестве решения задачи.

Пусть переходный процесс $y(t)$ может быть измерен в дискретном числе точек t_j, $1\leqslant j\leqslant m$, $t_1=0$, $t_m=T$ интервала $0\leqslant t\leqslant T$. Тогда численный метод решения задачи может быть следующим: заменим минимизацию S по непрерывному интервалу $0\leqslant t\leqslant T$ минимизацией на дискретном множестве точек:

$$S_1=\max_{1\leqslant j\leqslant m}\ \max_{1\leqslant i\leqslant 2}|x_i(t_j)-y_i(t_j)|.$$

В S_1 значения $x_i(t_j)$ будем находить численным интегрированием системы дифференциальных уравнений с заданной точностью ε. При фиксированном значении k величина S_1 есть функция k, т. е. $S_1=S_1(k)$. Приближенным значением k_* к точному k будем считать такое значение, которое минимизирует $S_1(k)$:

$$\min_{k_0\leqslant k\leqslant k_1} S_1(k).$$

Обозначим алгоритм этой задачи Z0A0. Проект алгоритма на нулевом уровне представлен на рис. 2.13.

Входные данные: k_0, k_1, x_1^0, x_2^0, t_j $(1\leqslant j\leqslant m)$, $y_i(t_j)$.

Выходные данные: k_*.

Обращаясь к библиотеке (см. гл. 12), находим, что среди базовых алгоритмов имеется два, которые можно использовать в проекте алгоритма, а именно: B5A0 — минимизация функции $S_1(k)$, B6-A0 — интегрирование системы дифференциальных уравнений (2.2.1). Опишем действия алгоритма Z0A0, необходимые для перевода входных данных задачи в выходные.

1) Вычисление $\min S_1(k)$ блоком B5A0 и определение k_*. Но для блока B5A0 требуется иметь алгоритм вычисления по значению k значения $S_1(k)$. Таким образом, первый уровень проекта можно представить в форме рис. 2.14. Здесь алгоритм B5A0 имеет в своем составе еще не разработанный блок вычисления значения $S_1(k)$. Обозначим этот блок S1K0. Тогда второй уровень проекта алгоритма можно представить в форме рис. 2.15. Для представления блока S1K0 заметим, что в его действие входит:

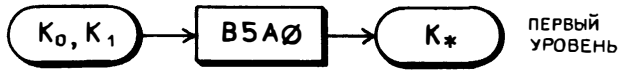

Рис. 2.14

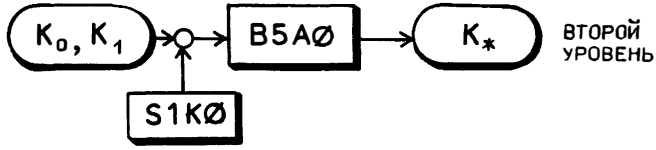

Рис. 2.15

2) Интегрирование системы уравнений (2.2.1) блоком B6A0, определение $x_i(t_j)$, $1 \leqslant i \leqslant 2$, $1 \leqslant j \leqslant m$.

3) Вычисление S_1 при фиксированном k.

Третий уровень проекта, представленный на рис. 2.16, есть блок-схема алгоритма S1K0. Здесь непомеченный блок производит следующие вычисления:

$$S_1 = \max_{1 \leqslant j \leqslant m} \max_{1 \leqslant i \leqslant 2} |x_i(t_j) - y_i(t_j)|.$$

Но для блока B6A0 требуется иметь алгоритм вычисления правых частей дифференциальных уравнений и алгоритм определения значений $x_i(t_j)$ (см. гл. 12). Обозначим соответствующие алгоритмы через F0A0 и F0A1 соответственно, непомеченный блок рис. 2.16

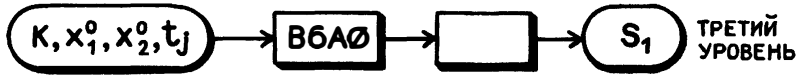

Рис. 2.16

обозначим F0A2. Тогда четвертый уровень проекта можно представить в форме рис. 2.17. Заметим, что алгоритмы четвертого уровня могут быть представлены с помощью базовых логических схем, базовых алгоритмов и элементарных арифметических действий над числами (см. рис. 2.11). Таким образом, проект рассматриваемого алгоритма исчерпывается пятью уровнями проектирования. В качестве примера алгоритма пятого уровня приведем блок-схему F0A2 (рис. 2.18). Вычисление S_1 и S_2 на этой схеме осуществляется блоком A0B5 (см. гл. 12).

Итогом проектирования алгоритма является структурно-логическая схема — объединение схем на рис. 2.13—2.18.

Теперь программирование алгоритма состоит в программировании:

1) обращений к базовым вычислительным блокам;

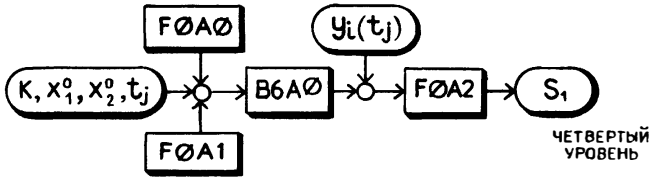

Рис. 2.17

2) базовых логических схем;
3) элементарных арифметических действий.

Проектирование алгоритма по указанным выше правилам будем называть структурной алгоритмизацией. Для дальнейшего изучения рассмотренных в этом пункте вопросов следует обратиться к [1].

● 2.3. Поток данных

После того как алгоритм спроектирован, т. е. построена его структурно-логическая схема, необходимо описать поток данных, проходящих через него. Основные алгоритмы вычислительной математики имеют как на входе, так и на выходе данные в виде чисел. Внутри алгоритма количество данных может изменяться, часть из них может запоминаться для дальнейших вычислений, часть — затерта записью на их место других данных. Внутри алгоритма возможна организация ввода и вывода данных. Если учесть, что размер памяти ЭВМ, имеющийся в распоряжении пользователя, ограничен определенной величиной, то станет понятной важность описания прохождения данных (потока данных) через алгоритм.

2.3.1. Карта данных алгоритма. Полное описание потока данных алгоритма — это представление всех данных на структурно-логической схеме алгоритма во всех точках входа в вычислительные блоки и выхода из них. Пусть имеется схема алгоритма (рис. 2.19), где цифрами от 1 до 10 обозначены все точки входа и выхода вычислительных блоков данного алгоритма. В каждой точке 1—10 необходимо указать все данные, которые имеет алгоритм в этой точке для схем без повторений. Для схем с повторениями указываются данные в точке *7* при входе в схему, в точке *8* — перед выходом. В каждой точке составляется карта данных алгоритма. Таким образом, алгоритм, приведенный на рис. 2.19, имеет 10 пронумерованных карт.

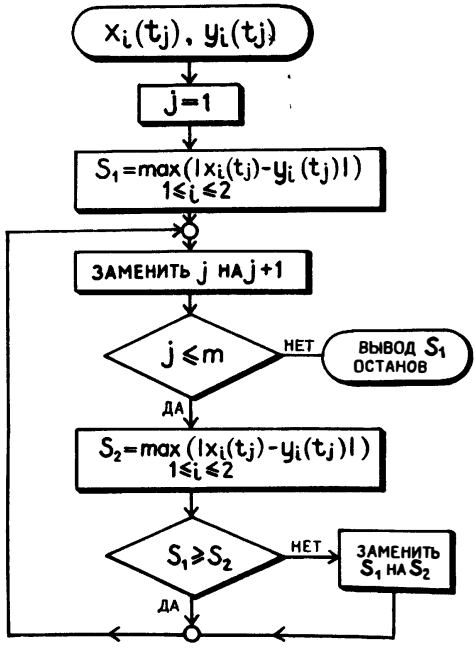

Рис. 2.18

Алгоритм, приведенный на рис. 2.19, решает следующую задачу: в точке *1* вводится матрица A, вектор b. В точке *2* имеем решение системы линейных уравнений

$$Ax=b,$$

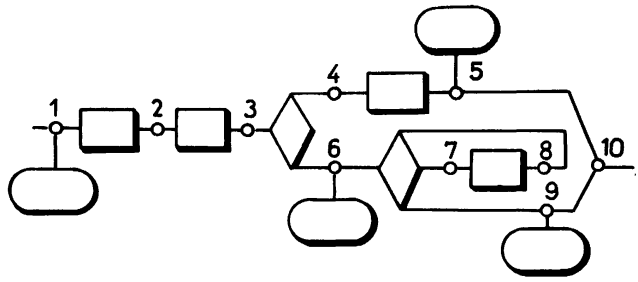

Рис. 2.19

поэтому в карте *2* добавляется вектор x к входным данным. В точке *3* вычисляется значение нормы $\|x\|=\max|x_i|$. Далее проверяется условие: если норма $\|x\|<1$, то переход на *4*, если $\|x\|\geqslant 1$, то на *6*. Пусть $\|x\|<1$; тогда в точке *5* имеем сумму вектора правых частей и решения $z=x+b$. Поэтому в карте *4* добавляется вектор z. Если $\|x\|>1$, то переход на *6*, где вводится вектор y. Схема повторений вычисляет скалярное произведение векторов x и y:

$$q=\sum_{i=1}^{50} x_i y_i.$$

Поэтому перед выходом из схемы повторений кроме данных карты *7* имеем еще число q. Полный набор всех карт алгоритма следующий.

Карта 1. Двумерный массив вещественных чисел — матрица

$$A_{ij},\quad 1\leqslant i\leqslant 50,\quad 1\leqslant j\leqslant 50,$$

одномерный массив вещественных чисел b_i, $1\leqslant i\leqslant 50$.

Карта 2. A_{ij}, b_i, одномерный массив вещественных чисел x_i, $1\leqslant i\leqslant 50$.

Карта 3. A_{ij}, b_i, x_i, $\|x\|$.

Карта 4. Совпадает с картой 3.

Карта 5. A_{ij}, b_i, x_i, одномерный массив вещественных чисел z_i, $1\leqslant i\leqslant 50$.

Карта 6. A_{ij}, b_i, x_i, одномерный массив вещественных чисел y_i, $1\leqslant i\leqslant 50$.

Карта 7. Совпадает с картой 6.

Карта 8. A_{ij}, b_i, x_i, y_i — вещественное число q.

Карта 9. Совпадает с картой 8.

Карта 10. Совпадает с картой 5 или 9.

Поясним применение термина «карта» к способу описания данных. Во-первых, данные можно размещать на картах, представляющих память ЭВМ, графически, а не давать словесного описания, как это сделано выше, тогда термин «карта» адекватен способу описания. Во-вторых, термин «карта памяти» применяется при описании распределения памяти ЭВМ, необходимого для выполнения программы алгоритма.

Таким образом, карта данных алгоритма играет ту же роль, что и карта памяти для программы алгоритма, но появляется на более ранней стадии решения задачи (на уровне алгоритма).

Карта данных алгоритма позволяет: 1) осуществить контроль структурно-логической схемы алгоритма; 2) документировать алгоритм так, чтобы в нем мог разобраться каждый, а не только тот, кто его спроектировал.

Приведем простой пример контроля схемы алгоритма, приведенного на рис. 2.19, описанием потока данных. Существуют распространенные вычислительные блоки решения систем линейных уравнений $Ax=b$, которые вектор решения x записывают на место вектора правых частей (затирают b). Если применить такой блок, то в точке *2* будем иметь следующую карту.

Карта 2. A_{ij}, одномерный массив x_i.

Следовательно, ветвь алгоритма по точкам *3—4—8* уже не может быть реализована, если $\|x\|<1$.

● 2.4. Структура программ

Процесс создания команд для ЭВМ, следуя которым ЭВМ выполнит алгоритм, называется *программированием алгоритма*, короче — *программированием*.

Программа — это последовательность команд для ЭВМ.

Считая, что этапом, предшествующим программированию, является проектирование алгоритма и описание потока данных, можно утверждать, что программирование — это перевод структурно-логической схемы алгоритма и потока данных на язык машинных команд — получение программы.

Программу создает пользователь ЭВМ, выполняет программу (обрабатывает команды) — процессор.

Выполняемая программа называется *вычислительным процессом*, короче — *процессом*.

Программы существовали задолго до появления ЭВМ, например музыкальная партитура — это программа для музыканта, кулинарный рецепт — программа для повара. Можно обнаружить много общих свойств между любыми программами, выполняемыми одним процессором.

1) Команды выполняются последовательно, если нет других указаний, начиная с первой до последней. Указания могут нарушить последовательность выполнения программы, например, требованием повторить часть музыкального произведения.

2) Процесс должен иметь результат — выведенные на терминал числа и символы, звуки музыки.

3) Часто перед командами программ располагается описание объектов, которые программа обрабатывает. В кулинарных рецептах дается список продуктов для приготовления блюда. В некоторых языках программирования необходимо описать данные перед командами.

4) Часто программы строятся так, что автор не знает, какие вычисления будет выполнять процессор,— это зависит от обрабатываемых данных, но всегда должен быть указан критерий выбора пути вычислений.

5) В программах применяются указания на необходимость повторения команды или последовательности команд более одного раза. В этом случае указывается либо точное число повторений (например, 100 раз), либо критерий, который зависит от процесса (повторять, пока точность вычислений не достигнет заданной $\varepsilon = 10^{-6}$).

Спецификой программ для ЭВМ является тот важнейший факт, что любая программа может быть построена с помощью следующих четырех базовых конструкций:

1. Последовательность команд.
2. Принятие решения (альтернатива).
3. Повторение (цикл).
4. Процедура.

Первые три конструкции имеют тот же смысл, что и в **2.2,**— это базовые логические схемы, где вычислительный блок есть группа команд или команда (см. рис. 2.5—2.7).

Процедура — это группа команд, которая заменяется одной командой. Например, кулинарная книга содержит процедуру приготовления бульона, а затем ссылается на эту процедуру в каждом рецепте, требующем бульона.

Процедуры в программировании ЭВМ уменьшают размеры программы и придают ей иерархическую структуру «сверху вниз» — от сложного к простому.

В архитектуре алгоритма вычислительный блок можно рассматривать как процедуру.

2.4.1. Программирование структуры блоков алгоритма. Рассмотрим общие вопросы программирования алгоритма, спроектированного по принципам структурной алгоритмизации **2.2** на уровне готовых вычислительных блоков. Иными словами, будем считать, что весь алгоритм состоит из уже написанных процедур.

Тогда процесс программирования алгоритма состоит из программирования:

1) ввода и вывода данных;
2) обращения к процедурам;
3) программирования трех логических схем.

Для программирования применяются различные языки, которые облегчают трудоемкий процесс перевода алгоритма в машинные команды. Однако следует помнить, что алгоритм, записанный на языке программирования, отличном от языка машинных команд, не может быть выполнен на ЭВМ непосредственно. Он должен быть переведен с этого языка специальной программой ЭВМ (например, транслятором для языка фортран, паскаль и т. п.).

Профессиональные программисты, владеющие несколькими языками программирования, твердо уверены, что лучше всего нужно

владеть русским языком. Если пользователь может изложить на русском языке структуру алгоритма так, чтобы другой пользователь смог ее понять, то у него не будет проблем с программированием на любом языке.

В различных языках программирования существуют свои правила записи элементов программирования. Поэтому, чтобы написать программу на конкретном языке, необходимо в следующих ниже программах заменить элементы 1)—3), написанные на русском языке, их соответствующими эквивалентами.

Для языка фортран соответствующие эквиваленты описываются в гл. 3.

Напишем программу алгоритма, приведенного на рис. 2.19, где схему повторений заменим одной процедурой вычисления скалярного произведения (рис. 2.20).

Нумерацию точек входа, выхода сохраним (рис. 2.19).

1. Ввести вещественные матрицу A_{ij}, вектор b_i, $1 \leqslant i, j \leqslant 50$. Обратиться к процедуре решения системы линейных уравнений $Ax = b$, получить вектор x_i, $1 \leqslant i \leqslant 50$. Обратиться к процедуре вычисления $\|x\| = \max|x_i|$, $1 \leqslant i \leqslant 50$, получить $\|x\|$.

3. Если $\|x\| > 1$, то выполнять 6, иначе:

Обратиться к процедуре вычисления вектора $z = x + b$, получить z.

5. Вывести A, b, x, $\|x\|$, z.

6. Ввести вещественный вектор y_i, $1 \leqslant i \leqslant 50$. Обратиться к процедуре вычисления $q = \sum_{i-1}^{50} x_i y_i$, получить q.

9. Вывести A, b, x, $\|x\|$, q.

10. ...

В приведенной выше программе использованы команды программирования ввода, вывода (1., 5., 6., 9.), обращения к процедурам (строки, начинающиеся словом «обратиться»), программирование последовательной логической схемы между 1. и 3., программирование альтернативной логической схемы — команды между 3. и 10. Причем первой ветви (блоку 1) соответствуют команды между 3. и 5., второй ветви (блоку 2) — команды между 6. и 9.

Альтернативная схема (рис. 2.21) общего вида программируется следующим образом. Если условие выполняется, то вычисления:

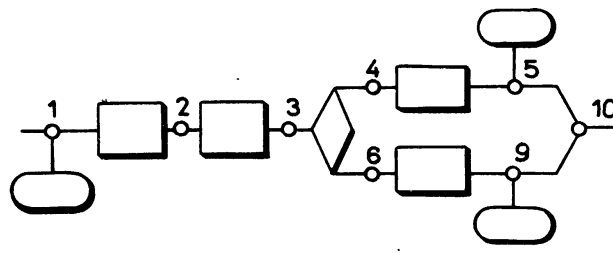

Рис. 2.20

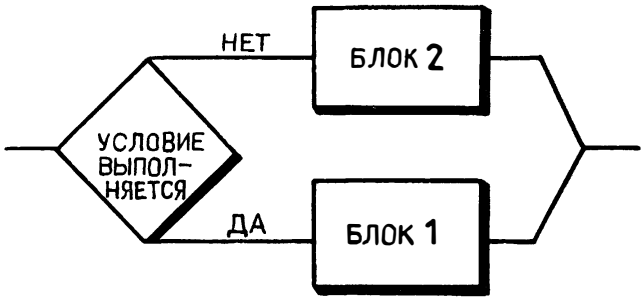

Рис. 2.21

на блоке 1, иначе:
на блоке 2.

В записи команд вычислительного блока 2 для наглядности следует все команды смещать на несколько позиций вправо относительно блока 1.

Схема повторений с проверкой условия до выполнения блока (рис. 2.22, *а*) программируется следующим образом:

Пока условие выполняется,
вычисления на блоке.

Программирование схемы (рис. 2.22, *в*) с проверкой условия после выполнения блока программируется следующим образом:

Выполнять блок, пока
не выполнится условие.

И хотя вторая логическая схема в) может быть сведена к схеме а), во многих языках имеются адекватные средства и для той, и для другой схемы.

Заканчивая описание программирования на уровне готовых вычислительных блоков или процедур, можно отметить, что на этом уровне следует на конкретном языке программирования (в этой книге — фортран) реализовать следующие базовые команды, упомянутые на русском языке:

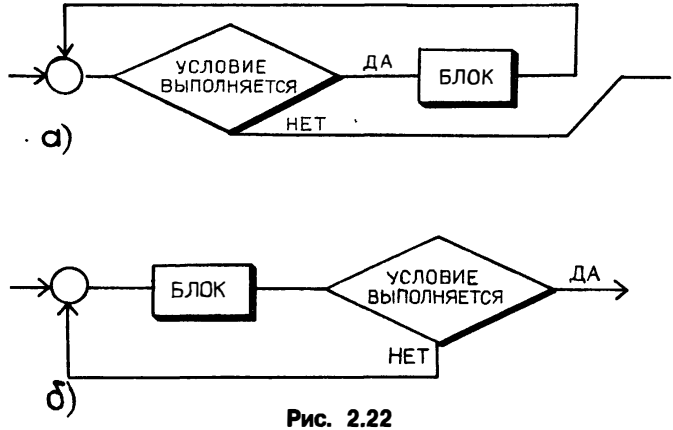

Рис. 2.22

Ввести ...
Вывести ...
Обратиться к процедуре ...
Если ... то, ... иначе
Пока ..., ...

Программирование структурированного алгоритма с использованием только базовых логических схем и базовых команд является характерной чертой *структурного программирования.*

2.4.2. Описание данных. В алгоритмах вычислительной математики данные, представляющие различные совокупности целых, вещественных и комплексных чисел, являются сравнительно простыми объектами. Некоторые алгоритмические языки, например фортран, позволяют перед командами не давать описания данных (использовать неявное описание; см. гл. 3), другие, например паскаль, требует обязательного описания.

Будем придерживаться строгого соблюдения правила: *все данные, которые обрабатывают вычислительные блоки алгоритма, должны быть описаны до первой команды.*

Очевидно, что константы алгоритма могут иметь лишь только строго предписанный данному типу вид и не описываться (см. гл. 3).

Будем также строго соблюдать следующее правило:

все константы должны быть описаны так же, как и переменные; значения им должны быть присвоены до первого их употребления в алгоритме.

Более жесткие, чем позволяет фортран, правила работы с данными, принятые нами, дают преимущества при поиске ошибок и их устранении, при распределении памяти ЭВМ, а в конечном итоге повышают эффективность работы на ЭВМ.

Переменные и константы в алгоритмах вычислительной математики описываются в виде скалярных данных и в виде массивов данных.

Скалярные данные: целого типа; вещественного типа; вещественного типа с двойной точностью; комплексного типа.

Примерами скалярных данных являются:

а) целого типа — число повторений цикла $N=20$;

б) вещественного типа — значение скалярного произведения приведенного выше алгоритма;

в) вещественного типа с двойной точностью — то же число q, что в п. б), только с увеличенной длиной мантиссы (см. гл. 1);

г) комплексного типа — корни z_1, z_2 квадратного уравнения с дискриминантом, меньшим нуля.

Массивы данных — это совокупности скалярных данных одного типа с указателем размерности массива.

Примерами одномерных массивов данных служат:

а) одномерные массивы x, b, z алгоритма (см. рис. 2.19) — векторы x_i, b_i, z_i, $1 \leqslant i \leqslant 50$;

б) массив значений функции $f(x)$, вычисленных в точках x_i, $1 \leqslant i \leqslant 100$ (обозначается f_i).

Примерами двумерных массивов данных являются:

а) двумерный массив А алгоритма (см. рис. 2.19) — матрица A_{ij}, $1 \leqslant i,\ j \leqslant 50$;

б) массив значений функции двух переменных $f(x, y)$, вычисленных в точках плоскости (x, y), (x_i, y_j), $1 \leqslant i,\ y \leqslant 100$, (обозначается $f_{i,j}$).

Примерами k-мерных массивов данных являются значения функции k переменных $f(x_1, x_2, .., x_k)$, вычисленные в точках $x_{1,i_1}, x_{2,i_2}, ..., x_{k,i_k}$, $1 \leqslant i_1 \leqslant N_1$, $1 \leqslant i_2 \leqslant N_2$, ..., $1 \leqslant i_k \leqslant N_k$ (обозначается $f_{i_1 i_2, ..., i_k}$).

В различных языках программирования формы описания данных имеют некоторые различия, но в основном это форма следующая: указатель типа и список данных этого типа. Например,

Целый i, j, k,

Вещественный q,

Вещественный с двойной точностью r,

Комплексный z,

Массив вещественный $a_{i,j}$, $1 \leqslant i, j \leqslant 50$,

Массив целый l_i, $1 \leqslant i \leqslant 20$.

Такое описание дает возможность подсчитать необходимый объем памяти ЭВМ, который будет занимать данные алгоритма. Для приведенного примера с учетом представления чисел в памяти (см. гл. 1) получаем, что для i, j, k необходимо 6 байт, для q — 4 байт, r — 8 байт, z — 8 байт, $a_{i,j}$ — 10 000 байт, $f_{i,j,k}$ — 4000 байт, l_i — 40 байт; итого — 14 066 байт.

2.4.3. Программирование вычислительных блоков для процедур. Как уже отмечалось выше, процедура может рассматриваться как готовая программа, к которой лишь нужно уметь обратиться. Однако многие из процедур для своей работы требуют наличия некоторых программ. Их обязан подготовить пользователь. Так, например, для процедур вычисления интегралов необходимо написать программу вычисления подынтегральной функции (блок А4А0), для процедур численного интегрирования обыкновенных дифференциальных уравнений (блок В6А0) необходимо написать программу вычисления правых частей уравнений и т. п. Такие программы часто называют внешними.

Кроме того, процедуры для своей работы могут требовать наличия ячеек памяти для промежуточных результатов. Поэтому, составляя список описания данных алгоритма, необходимо включать в него и те данные, которые требуют для своей работы все процедуры. Эта информация содержится обычно в описании соответствующих процедур.

Таким образом, даже когда спроектированный алгоритм содержит готовые процедуры, программирование вычислительных блоков оказывается актуальной задачей.

Повторяя процесс проектирования алгоритмов этих блоков «сверху вниз», мы в конце концов придем к необходимости программировать только элементарные вычислительные блоки.

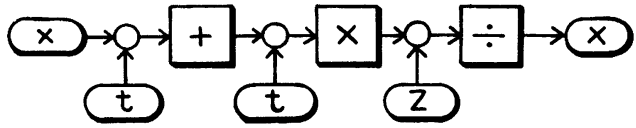

Рис. 2.23

2.4.4. Программирование элементарных вычислительных блоков. Алгоритмические языки программирования, ориентированные на вычислительные задачи, позволяют программировать элементарные блоки, объединяя четыре арифметические операции в конструкции, по форме близкие к математической записи.

Переменные можно обозначать буквами; согласно принятым нами правилам, и константы следует обозначить буквами, например x, y_i, z, a, b.

Каждой переменной и константе в соответствии с описанием отводятся ячейки памяти ЭВМ, которые обозначают теми же буквами.

Введем понятие оператора присваивания, который обозначим знаками $:=$, чтобы отличить от математически привычного знака $=$. Запись

$$x:=(x+t)\,x*t\div z$$

соответствует схеме, приведенной на рис. 2.23, и отвечает следующим действиям ЭВМ: из ячеек x и t выбирается содержимое и направляется на блок суммирования; полученный результат и содержимое ячейки t направляются на блок умножения; полученный результат и содержимое ячейки z направляются на блок деления; полученный результат посылается в ячейку x. Отличие оператора присваивания от математического равенства теперь легко понять. Кроме вычислений правой части выражения оператор присваивания засылает результат в ячейку, которая обозначается левой частью оператора, при этом затирается старое содержимое ячейки x.

Программирование любого элементарного блока выполняется с помощью трех базовых логических схем, которые используют следующие базовые команды, называемые операторами: ввода, вывода, альтернативы, цикла, присваивания, записи чисел, скобок, знаков арифметических операций и меток операторов.

Глава 3

ПРОГРАММИРОВАНИЕ НА ФОРТРАНЕ

● 3.1. Введение

3.1.1. История фортрана. История языков программирования тесно связана с историей развития вычислительной техники. За прошедшие с 40-х годов десятилетия были созданы тысячи языков программирования высокого уровня. Десятки из них нашли применение в узких направлениях применения ЭВМ. Большинство языков остались достоянием их создателей, и только некоторые стали широко распространенными и играют роль гораздо большую, нежели создание программ для ЭВМ. Такие языки программирования, как фортран, алгол, си, пролог, паскаль, бэйсик, позволяют создавать программы, которые можно выполнить на различных ЭВМ, обмениваться программами и помогают общаться специалистам в разных предметных областях и странах.

Среди языков высокого уровня в настоящее время нет языка, сравнимого с фортраном по долголетию, распространенности и эффективности получаемых программ для вычислительных задач научно-технического содержания. На фортране написано такое количество программ, что трудозатраты, по-видимому, можно оценить миллионами человеко-лет. Ни один производитель вычислительной техники с ориентацией на научно-техническое применение не может игнорировать наличие в мире огромного накопленного программного продукта на фортране. Все ЭВМ такого типа оснащаются трансляторами с фортрана.

Первая версия языка была разработана фирмой «IBM» в конце 50-х годов. Эта версия была ориентирована на устройства ввода с перфокарт и несет отпечаток архитектуры вычислительной техники 60-х годов. В первые годы отсутствовало стандартное определение фортрана, что отрицательно сказывалось на переносимости программ с одной ЭВМ на другую, имеющую другой транслятор. Положение исправилось в 1966 г., когда был введен стандарт языка, известный как фортран 66. В результате улучшилась мобильность (переносимость) фортран-программ. Программа, написанная в рамках этого стандарта, без изменений (или с небольшими изменениями) может выполняться на различных ЭВМ, имеющих соответствующие трансляторы.

Трансляторы с фортрана для разных типов машин обычно дополняются расширенными возможностями по сравнению со

стандартом. Однако следует всегда иметь в виду, что использование нестандартного фортрана может привести к трудностям при переносе уже написанных программ на другую ЭВМ.

За 10 лет (с 1966 по 1977 г.) фортран завоевал среди пользователей ЭВМ большую популярность, сфера его приложений стала расширяться за пределы научно-технических расчетов. В это же время произошла смена поколений вычислительной техники. Технология ввода-вывода с перфокартами утратила свои позиции, получили широкое распространение видеотерминалы. В результате появилась потребность в модификации фортрана. Была разработана новая версия языка, расширяющая возможности фортрана, которая учитывала новые приложения и вычислительную технику. В 1977 г. был введен стандарт языка, известный как фортран 77. В программном обеспечении ЭВМ, как правило, имеются трансляторы с обоих языков. Причем старая версия фортрана практически является частью (подмножеством) новой версии. Поэтому написанные ранее программы без существенных изменений могут транслироваться новыми трансляторами.

Следующий десятилетний цикл обновления фортрана связан с учетом теории современных языков программирования, принципов структурного программирования. Новый стандарт фортрана, называемый фортран 8х, выражает модульную структуру языка и утверждает принципы его развития. Эта концепция состоит в том, что язык должен состоять из нескольких модулей, важнейшие из которых следующие:

модуль ядра фортрана;
модуль архаизмов;
модуль расширений.

В модуль расширений включаются средства программирования для ЭВМ с новой архитектурой. Например, широкое распространение векторных процессоров потребовало включить в фортран средства работы с числовыми массивами как с отдельными числами — так называемое векторное расширение фортрана.

При всех обновлениях фортрана сохраняются два основных его достоинства: эффективность программ и простота применения.

3.1.2. Обучение программированию на фортране. Цель настоящей главы — ознакомиться с основами техники программирования вычислительных задач на фортране. Методика обучения программированию включает знакомство с минимальным фортраном в объеме п. 3.2. Для некоторых пользователей это и максимум необходимых знаний. Изложение за небольшим исключением ведется на уровне фортрана 66 без дальнейших оговорок.

В минимальном фортране оставлены только те средства языка, без которых нельзя обойтись в несложной вычислительной задаче. Минимальный фортран можно было бы еще сократить, оставаясь верным структурному программированию (см. гл. 2). Это не сделано по той причине, что возможны трудности в понимании

текста чужих программ. С первых шагов обучения программированию необходимы работа на ЭВМ и решение учебных задач с использованием каждого изученного оператора фортрана.

После изучения минимального фортрана следует освоить приемы программирования основных блоков вычислительной математики (см. **3.3**). Умение запрограммировать достаточный набор таких блоков позволит решать уже более сложные задачи. Следуя идее структурной алгоритмизации (см. гл. 2), нужно сложный алгоритм представить в виде структуры простых блоков, а затем использовать программы, написанные для блоков.

Следующий этап обучения наступает тогда, когда пользователя уже не удовлетворяет «какая-нибудь» программа решения задачи. Возникает желание или необходимость написать в некотором смысле оптимальную программу (по быстродействию, с минимальными затратами памяти и т. п.). Некоторые приемы оптимизации в программировании изложены в **3.4**.

Наконец, появляется достаточный опыт, который позволяет применять широкие возможности фортрана. Некоторые расширения минимального фортрана, а именно на уровне фортрана 77, излагаются в **3.5**.

Важным моментом обучения программированию на всех этапах является стремление написать простую по стилю программу. И хотя понятие простоты, как и красоты, определить довольно трудно, простая программа отличается тем, что ее работу можно понять в отсутствие автора. Такому стилю программирования отвечает концепция структурного программирования. Этот стиль не связан с конкретным языком. В более поздних, чем фортран, языках (например, паскале) заложены эти концепции. На фортране есть возможность писать плохие по стилю программы. Поэтому с начального этапа обучения следует избегать «дурных привычек» в программировании.

● 3.2. Основы программирования на фортране

3.2.1. Символы. При записи программ на фортране используются следующие символы:

1) Прописные буквы латинского алфавита от A до Z;
2) Цифры от 0 до 9;
3) Специальные символы;

— пробел
= — знак равенства
\+ — плюс
− — минус
* — звездочка
/ — косая черта
) — правая скобка
(— левая скобка

, — запятая
. — десятичная точка
′ — апостроф
” — кавычки
$ — денежный символ
: — двоеточие.

Прописные буквы русского алфавита и другие символы могут встречаться в фортране только как часть текстовой константы или в комментариях, о которых сказано ниже.

Из символов языка образуются ключевые слова (например, IF (если), DO (делать), GO TO (идти к) и т. п.), имеющие строго определенный смысл, и слова пользователя — идентификаторы. *Идентификаторы* — это последовательность (не более шести символов) букв и цифр, начинающаяся с буквы.

Примеры.

1) Допустимые идентификаторы

A15, BETA, INDMAS, A1B2C

2) Недопустимые идентификаторы

C*B3 (содержит *)
5ALPHA (начинается с цифры)

Поскольку слова пользователя придется набирать на клавиатуре, целесообразно их выбирать по возможности короче.

3.2.2. Константы и переменные. *Константы* представляют собой неизменяемую величину в процессе вычислений. Рассмотрим следующие типы констант: целые, вещественные, вещественные с двойной точностью, комплексные, логические, текстовые.

Целая константа имеет следующий вид:

SN1N2...

где N1N2 — последовательность десятичных цифр, а S — знак числа. Если константа не имеет знака, она считается положительной. Допустимая область значений для целых констант (в ЭВМ с 16-битовым словом) — от — 32768 до + 32767.

Примеры.

1) Правильная запись целых констант

0
−1514
+31539
275
037

2) Неправильная запись целых констант

50327 (слишком велика)

3.14 (присутствует десятичная точка)

Вещественная константа может быть записана двумя способами.

1. Вещественная константа без порядка имеет вид

SN1N2.N1N2

где N1N2 — последовательность десятичных цифр, а S — знак числа.

2. Вещественная константа с порядком записывается в виде

K1EK2

где K1 — вещественная константа без порядка, K2 — порядок (однозначная или двузначная целая константа).

Значение вещественной константы находится в пределах от $0{,}29 \cdot 10^{-38}$ до $1{,}7 \cdot 10^{+38}$ (в ЭВМ с 32 разрядами под запись числа, из которых 24 бит отводятся под мантиссу).

Примеры.

1) Правильная запись вещественных констант

3.14159
3734.
−0.2134
28.E31
2.E−5

2) Неправильная запись вещественных констант

$25. 5 (специальный символ)
41.3E51 (слишком велика)

Вещественные константы удвоенной точности задаются в виде вещественной константы с порядком. Вместо буквы E используется буква D.

Пример.

+131.5D+2

Этот тип константы занимает в памяти 64 разряда. Точность представления числа с использованием 32 разрядов — порядка 7 десятичных цифр после запятой (0,9345671), с использованием 64 разрядов — 17 десятичных цифр (0,93456718013542166). Диапазон изменения констант этого типа такой же, как у вещественных констант обычной длины.

Комплексная константа представляет собой пару вещественных констант, разделенных запятой и заключенных в скобки. Первая вещественная константа представляет действительную часть комплексного числа, а вторая константа — мнимую часть. Комплексная константа имеет следующий вид:

(R1, R2)

где R1, R2 — вещественные константы.

Примеры.

1) Константа (25.236,—1.3789) соответствует

25,536 — i*1,3789, где i — мнимая единица.

2) Константа (+17567.E−4,0.) соответствует 1,7567.

Логическая константа может принимать только два значения:

.TRUE. — «истина»
.FALSE. — «ложь»

Ограничивающие точки являются обязательной частью каждой логической константы. Значения логических величин .TRUE. и .FALSE. представляются знаковым разрядом ячейки. Нуль в этом разряде соответствует .TRUE., а единица соответствует .FALSE.. Остальные разряды ячейки нулевые.

Текстовая константа — это последовательность символов, которой предшествуют указатель их числа и буква H, так называемая холлеритова константа:

NHC1C2 ... CN

Примеры.

17HПРОГРАММА ПЕТРОВА
5HSTAR:

Максимальное число символов в константе — 255. Текстовые константы удобно применять для сопровождения ввода и вывода чисел, таблиц чисел, поясняющей текстовой информации.

Текстовая константа может быть определена без указателя длины. Такая константа имеет вид

'C1C2 ... CN'

(последовательность символов заключается в апострофы).

Примеры.

'ПРОГРАММА ПЕТРОВА'
'STAR:'

Переменная представляет собой изменяемую величину в процессе вычислений. Переменная обозначается идентификатором.

Пример.

X0, X1, EPS1.

Переменные разделяются по типу, так же как и константы, на целые, вещественные и т. д. Тип переменной может быть задан неявно. Если идентификатор начинается с букв

I, J, K, L, M, N,

то переменная целого типа, с остальных букв — вещественного:

IRI, KAMA, N — целый тип,
R1, AMA, AN — вещественный тип.

Кроме того, тип переменной может быть задан явно операторами описания типа, которые приводятся ниже. Тип переменной нужно установить до начала употребления переменной в программе, так как транслятор в соответствии с типом распределяет память ЭВМ. Например, для переменной целого типа отводится одно

слово — 16 разрядов, для вещественного типа — 32 разряда, для комплексного — четыре слова — 64 разряда.

Неявный способ задания типа переменной называется иногда соглашением по умолчанию. Мы не будем в практике программирования использовать неявный способ, хотя фортран это и позволяет и его можно встретить в текстах программ.

Явное описание типа переменных заставляет провести контроль употребляемых переменных, проследить за распределением памяти, исключить двойное употребление переменных.

Явное описание осуществляется с помощью невыполняемых операторов: указателей типа переменных.

Программа на фортране состоит из последовательности операторов, которые делятся на выполняемые и невыполняемые.

Первыми невыполняемыми операторами, с которых следует начинать писать программу, являются указатели типа данных, встречающихся в вычислениях. Указатели типа следующие:

INTEGER	— целый тип,
REAL	— вещественный тип,
DOUBLE PRECISION	— вещественный с двойной точностью,
COMPLEX	— комплексный тип,
LOGIGAL	— логический тип.

Операторы описания типа отменяют соглашения по умолчанию, например

```
INTEGER A, B
REAL U, K
COMPLEX G, D, I
LOGICAL Q
```

Операторы описания типа должны предшествовать любому первому употреблению в программе переменных, которые они определяют.

3.2.3. Массивы. *Массив* — это последовательность переменных одного типа, обозначаемая идентификатором с индексами, заключенными в круглые скобки. Например, вектор u с вещественными компонентами u_1, u_2, ..., u_n представляется одномерным массивом.

U(I), $1 \leqslant I \leqslant N$.

Матрица A с компонентами — целыми числами $A_{i,j}$, $1 \leqslant i \leqslant N$, $1 \leqslant j \leqslant M$, представляется двумерным массивом

KA(I, J), $1 \leqslant I \leqslant N$, $1 \leqslant J \leqslant M$.

Значения индексов больше либо равны 1. Массив размещается в последовательно расположенных ячейках памяти. Размерность массива не более 3.

Массив, размерность которого больше 1, располагается так, что левый индекс меняется быстрее правого. Для матриц это соответствует тому, что матрица A хранится в памяти по столбцам

A(1, 1) A(1, 2) A(1, 3)
A(2, 1) A(2, 2) A(2, 3)
A(3, 1) A(3, 2) A(3, 3)

а именно: A(1, 1), A(2, 1), A(3, 1), A(1, 2), A(2, 2), A(3, 2), A(1, 3), A(2, 3), A(3, 3). Для обращения к конкретному элементу массива должны быть указаны или вычислены значения его индексов, например U(3) — третий элемент массива U, A(2, J) — элемент второй строки J-го столбца матрицы A (массива A). Значение J предварительно должно быть вычислено. Размер массива и тип его элементов в программе должны быть описаны до первого употребления его элементов с помощью операторов описания типа, где указываются максимальные значения индексов. Например,

```
REAL U(20)
```

означает, что в программе будет использован одномерный массив вещественных чисел u_i, $1 \leqslant i \leqslant 20$, или

```
INTEGER A(5, 7)
```

означает, что в программе будет использован двумерный массив целых чисел $A_{i,j}$, $1 \leqslant i \leqslant 5$, $1 \leqslant j \leqslant 7$.

Размер массива может быть также определен с помощью невыполняемого оператора DIMENSION по аналогии с вышеизложенным, например

```
DIMENSION U(20), A(5, 7)
```

Но чтобы описать массив A(5, 7) как массив целых чисел, необходимо применить еще оператор описания типа

```
INTEGER A(5, 7)
```

в противном случае соглашение по умолчанию устанавливает, что массив A состоит из вещественных чисел. Таким образом, если придерживаться строго описания типа всех переменных программ, то оператор DIMENSION оказывается лишним. В стандарте фортрана 8x этот оператор переведен в модуль архаизмов.

3.2.4. Арифметические выражения и библиотечные функции. *Арифметическое выражение* — это запись математической формулы с использованием констант, переменных, массивов, функций, соединенных знаками арифметических операций и скобками по правилам фортрана.

Знаки арифметических операций:

\+ сложение
\- вычитание
* умножение
/ деление
** возведение в степень.

Следует соблюдать следующие правила:

1) Нельзя опускать знак умножения:

AB не есть A*B

2) Нельзя подряд писать два знака операций:

нельзя писать A*−B, нужно A*(−B) или −A*B

Порядок вычислений — по старшинству операций, при равном старшинстве — по порядку слева направо.

Порядок старшинства:

1) Вычисление функций.
2) Возведение в степень.
3) Умножение и деление.
4) Сложение и вычитание.

Выражения в скобках вычисляются в первую очередь и в порядке записи слева направо; в случае вложения скобок вычисления начинаются с внутренних. В вычислении выражений целого типа при делении дробная часть отбрасывается, а не округляется. Например, значение выражения 1/3+2/3 равно 0, а не 1. В арифметических выражениях могут участвовать величины четырех типов: целые, вещественные, с двойной точностью, комплексные. Здесь они записаны в порядке возрастания ранга. Тип выражения такой, как у элемента с наибольшим рангом. Например, тип результата

2+2. вещественный
.2E−2/.3D−1 с двойной точностью

Приведем примеры арифметических выражений.

Математическая запись	*Запись на фортране*
$\frac{A+B}{C}$	(A+B)/C
$x^2+\sin y^2$	X**2+SIN(Y**2)

Для вычисления индексов массивов допускаются следующие арифметические выражения:

целая константа * целая переменная ± целая константа

Например,

V(I−1)+V(2*I+3)

В программе можно обратиться к библиотечным функциям фортрана, указав имя и аргументы в скобках после имени (как в приведенном выше примере — SIN(Y**2)). Аргументы при этом могут быть арифметическими выражениями. Пользователю не нужно писать программу вычисления таких функций, готовую (библиотечную) программу включит в нужном месте программы пользователя операционная система.

Приведем некоторые библиотечные функции (полный список см. в гл. 12).

Математическая запись	*Запись на фортране*
$\|x\|$	ABS(X)
$\sin x$	SIN(X)
e^x	EXP(X)

3.2.5. Арифметический оператор присваивания. Как уже отмечалось, программа представляет собой последовательный набор операторов. Операторы делятся на невыполняемые и выполняемые. С некоторыми невыполняемыми операторами мы уже познакомились (операторы описания типа переменных). Содержательная программа должна иметь по крайней мере один выполняемый оператор.

К выполняемым операторам относится *арифметический оператор присваивания*. Вид оператора присваивания

```
W=A
```

где W — идентификатор (имя) переменной, A — арифметическое выражение.

Оператор присваивания не эквивалентен математическому знаку равенства. Фактически этот оператор производит два действия:

1) вычисление арифметического выражения A;

2) присваивание значения арифметического выражения переменной.

Например, оператор

```
X=X+1
```

означает, что вычисляется выражение X+1 и прежнее значение переменной X заменяется (затирается) на новое значение X+1.

С помощью оператора присваивания можно преобразовывать целые величины в вещественные и наоборот. Так, если справа имеем выражение, состоящее из целых констант и переменных, а слева — наименование вещественной переменной, то вычисления производятся по правилам действий с целыми числами, а результат преобразуется в вещественную форму и присваивается переменной левой части.

Примеры арифметических операторов присваивания:

```
A(I)=S**2+Q+ALGOL(X)
F=F+B(I, K)*C(K, J)
```

Примеры неправильной записи:

```
X=A=3    (двойное присваивание не допускается)
SIN(X)=1 (в левой части функция — часть арифметического
          выражения, а должна стоять переменная)
```

3.2.6. Структура программы. Фортран-программа состоит из операторов. Каждый оператор печатается (набирается на клавиатуре дисплея) в строках длиной 80 символов. В строке не должно быть более одного оператора. Позиции символов в строке нумеруются слева направо начиная с 1:

```
1 2 3 4 5 6 7 8.....72.....80
```

Строки делятся на четыре поля:

позиции: 1 — 5, 6, 7 — 72, 73—80

Операторы фортрана печатаются в третьем поле в позициях 7—72, внутри этого поля расположение оператора произвольное.

Любой оператор фортрана может быть помечен меткой — десятичным целым числом. Метки помещаются в первом поле в позициях 1—5.

Если оператор не помещается в позициях 7—72 или для удобства желателен перенос оператора на следующую строку, то в 6-й позиции строки продолжения следует напечатать любой символ, кроме пробела и нуля. Первое поле строк продолжения должно быть пустым.

Четвертое поле (позиции 73—80) под запись операторов не используется. В эти позиции можно помещать, например, номера строк.

Если в первой позиции любой строки программы напечатана буква C

```
C       КОММЕНТАРИЙ
```

то такая строка транслятором полностью игнорируется, она рассматривается как комментарий. Текст комментария помещается в позициях

2—80. Например,

```
C       АВТОР ПРОГРАММЫ 2 ПЕТРОВ В. Н.
C       ВЕРСИЯ — 5, МАЙ 1990, ФОРТРАН 77
```

Правила заполнения строк, приведенные выше, распространяются только на операторы фортрана, но неприменимы к размещению данных. Ниже будет показано, что для данных используются все 80 позиций.

Последним оператором любой программы должен быть оператор

```
        END
```

Оператор

```
        STOP
```

прекращает выполнение программы; если оператор STOP отсутствует, то выполнение пограммы заканчивается на операторе END.

В качестве примера структуры фортран-программы приведем следующую программу:

```
C       ПРИМЕР СТРУКТУРЫ ФОРТРАН-ПРОГРАММЫ
C       ВЫЧИСЛЕНИЕ ЗНАЧЕНИЯ ФУНКЦИИ У(Х)
C       В ТОЧКЕ Х
        REAL X, Y
        X=0.3
10      Y=(SIN(X)+COS(X)-1.E-4*X**2)+
       *(ALOG(X)/(X+1))
        END
```

3.2.7. Операторы ввода — вывода. Вводить числа можно с помощью оператора присваивания, например

```
K=1
X=3.1415
```

В тех случаях, когда этот способ неудобен (программа используется многократно с различными исходными данными), применяются операторы ввода. Вывести результаты вычислений можно только с помощью оператора вывода.

Запись операторов ввода — вывода такова:

```
ввод   READ (U, F) A, B, ..., Z
вывод  WRITE (U, F) A, B, ..., Z
F      FORMAT (...)
```

Здесь U — номер устройства ввода — вывода, например дисплей, АЦПУ; A, B, ..., Z — список переменных, для которых нужно ввести (вывести) числовые значения; F — метка оператора

В скобках оператора FORMAT указывается спецификация (характеристика) позиций для каждой переменной списка A, B, ..., Z. Это необходимо делать, поскольку вводимые или выводимые числа могут быть целыми, вещественными, вещественными с двойной точностью; вещественные числа могут представляться в форме с десятичной точкой или с порядком.

Если вводится (выводится) целое число, то в операторе FORMAT указывается спецификация I

Im

где m — число позиций, отведенное этому числу.

Если вводится (выводится) вещественное число с десятичной точкой, то в операторе FORMAT указывается спецификация F

F$m.n$

где m — общее число позиций, отведенное этому числу, n — число позиций под дробную часть. Заметим, что одна позиция из общего числа отводится под знак числа, одна — под точку, одна — под нуль, поэтому следует указывать $m \geqslant n+3$.

Если вводится (выводится) вещественное число с порядком, то в операторе FORMAT указывается спецификация E

E$m.n$

где m — общее число позиций, отведенное этому числу, n — число позиций под дробную часть. Заметим, что одна позиция из общего числа отводится под знак числа, одна — под точку, одна — под нуль, одна — под букву E, одна — под знак порядка, две — под порядок, поэтому следует указывать $m \geqslant n+7$.

Для организации между числами p пробелов между спецификациями чисел указывается

pX

Рассмотрим примеры ввода. Ввод с терминала (номер устройства — 5) для переменных K, A, B(3) значений K = 37, A = 0,375, $B(3) = 25 \cdot 10^{-6}$ осуществляет следующий оператор ввода:

```
      READ (5, 10) K, A, B(3)
10    FORMAT (I2, F6.3, E9.2)
```

Заметим, что набор на клавиатуре терминала чисел 37; 0, 375; $25 \cdot 10^{-6}$ или другой тройки чисел производится на этапе запуска задачи на счет. Когда по ходу выполнения программы встретится этот оператор ввода, выполнение программы прервется и задача будет стоять в ожидании ввода. Тогда на клавиатуре набираются число по заказанному формату:

```
позиции:        12345678901234567
числа:          37 0.375 0.25E−06
спецификации: I2  F6.3     E9.2
```

При вводе встречающиеся внутри общего числа позиций пробелы считаются нулями, поэтому вводимые числа следует прижимать к правому краю своих позиций. Например, для другой тройки чисел К = 7, А = 7,5; $-1{,}1 \cdot 10^2$ на клавиатуре следует набрать

```
позиции:        12345678901234567
числа:           7  +7.5   −1.1E2
спецификации: I2   F6.3   E9.2
```

Если набрать по-иному, например

```
позиции:        12345678901234567
числа:          7    7.5    −1.1E2
спецификации: I2  F6.3    E9.2
```

то введутся числа К = 70; А = 7,5; $B(3) = -1{,}1 \cdot 10^{20}$

Прежде чем перейти к примерам вывода, следует отметить, что при выводе на устройства, требующие управления кареткой (дисплей, АЦПУ), первый символ выводимой информации не печатается, а используется для управления расположением строк по вертикали. Если этого не учитывать, то может произойти потеря информации. Поэтому при выводе в соответствующих операторах FORMAT всюду проставлены символы 2X, что означает два пробела, из которых один пробел интерпретируется как управляющий — переход на новую строку перед печатью. Управляющий символ задается в операторе формата символьной константой. Управляющие символы:

— переход на новую строку перед печатью,
0 — пропуск двух строк перед печатью,
1 — переход на новую страницу,
+ — текущая строка печатается на предыдущей.

Оператор вывода может не содержать списка выводимых чисел, а только ссылку на оператор FORMAT. Таким образом выводятся текстовые сообщения на терминал или АЦПУ. Например, вывод

```
      WRITE (5,1)
1     FORMAT ('0',5X, 'ТАБЛИЦА ЗНАЧЕНИЙ')
```

приведет к пропуску двух строк перед печатью на терминале; начиная с 6-й позиции будет напечатано: ТАБЛИЦА ЗНАЧЕНИЙ

Рассмотрим пример вывода значений переменных К, А, В(3) на АЦПУ (номер устройства — 6). Оператор вывода может быть следующим:

```
      WRITE (6, 3) K, A, B(3)
3     FORMAT (2X, I4, 2X, F6.2, 2X, E10.2)
```

Предположим, что значения этих переменных $К = -372$; $А = 3{,}7 \cdot 10^{-2}$; $В(3) = 2{,}5 \cdot 10^{-5}$. Тогда на АЦПУ выведется строка:

позиции:	123456789012345678901234 5		
числа:	−372	0.04	0.25E−04
спецификации:	I4	F6.2	E10.2

В примере показано, что при выводе значений переменных числа округляются так, чтобы они поместились в заказанный формат. Если число не может быть размещено в заданном формате, оно не выводится, а во всех его позициях печатается символ *. Если в примере значение $К = -3721$, то четырех позиций для вывода этого числа недостаточно, поэтому на АЦПУ в первых пяти позициях были бы следующие символы:

позиции:	123456
числа:	****
спецификации:	I4

При вводе и выводе спецификации в операторе FORMAT можно повторять, указывая число повторений перед этой спецификацией. Например, вывод значений трех переменных А, В, С можно осуществить с повторением:

```
      WRITE (5, 12) A, B, C
12    FORMAT (2X, 3E11.4)
```

На терминал будут выведены три числа в 33 позиции одной строки.

Если при вводе (выводе) все спецификации оператора FORMAT уже использованы, но не всем переменным списка присвоено значение (не все выведены), то использование спецификаций в нем повторяется начиная с первой. Например, вывод значений трех переменных А, В, С

```
      WRITE (5, 10) A, B, C
10    FORMAT (2X, E11.4)
```

осуществится на терминал в строки по 11 позиций в каждой.

Вывод текстовой информации совместно с числовой более предпочтителен, так как поясняет смысл выведенных значений. Например, вывод значений, А, В, С таким образом, чтобы перед числами стояли символы А=, В=, С=, осуществляется следующими операторами:

```
      WRITE (5, 1) A, B, C
1     FORMAT (2X, 'A=', E11.4, 'B=', E11.4, 'C=', E11.4)
```

Ввод чисел с текстовой информацией часто называют вводом с приглашением, который состоит в том, чтобы перед вводом

на терминале выводились символы тех переменных, для которых следует ввести значения, а затем на клавиатуре набиралось бы вводимое число. Это можно организовать при помощи четырех операторов, например

```
      WRITE (5,1)
1     FORMAT ('A=')
      READ (5,10) A
10    FORMAT (F6.3)
```

Если при вводе (выводе) необходимо перейти на следующую строку, то ставится знак/. Например, вывод в три строки предыдущего примера осуществляется операторами

```
      WRITE (5,1) A, B, C
1     FORMAT (2X, 'A=', E11.4/'B=', E11.4/'C=', E11.4)
```

На терминал будут выведены три строки:

A=...
B=...
C=...

Ввод (вывод) массива осуществляется разными способами. В простейшем варианте указывается имя массива (без индекса) в вводном (выводном) списке. Например, пусть необходимо ввести двумерный массив вещественных чисел

```
      REAL B(2,3)
```

Тогда операторы ввода могут быть, например, такими:

```
      READ (5,1)B
1     FORMAT (2F4.1)
```

Пусть необходимо ввести матрицу

$$B=\begin{pmatrix} 1 & -3 & 5 \\ 2 & 4 & -6 \end{pmatrix}.$$

При вводе с терминала согласно приведенным выше операторам (и вспоминая, что в памяти ЭВМ матрица размещается по столбцам) необходимо напечатать числа следующим образом:

```
позиции:       1 234 5 678
1-я строка:      1.0  2.0
2-я строка:     -3.0  4.0
3-я строка:      5.0 -6.0
```

Простейшая форма ввода (вывода) массива крайне неудобна, когда размерность массива может быть переменной. В этом случае применяется ввод (вывод) с помощью неявного цикла (см. п. 3.2.8).

Место оператора FORMAT в структуре программы строго не определено, он может находиться в любом месте, а не обязательно в следующей строке за оператором ввода (вывода). Заметим

также, что один оператор FORMAT могут использовать несколько операторов ввода (вывода), например

```
      READ (5, 1) A
1     FORMAT (2X, E13.6)
      WRITE (6, 1) B
```

3.2.8. Операторы передачи управления, циклы. Операторы выполняются в программе последовательно, однако часто необходимо в алгоритме изменять естественный процесс выполнения программы. Этой цели служат операторы передачи управления. Таких операторов несколько.

1) Оператор безусловной передачи управления

GO TO N

Здесь N — метка оператора, который должен выполняться следующим.

Пример.

```
      X=0.5
      GO TO 3
      Y=1.
      ...
3     X=0.5*X
      ...
```

В этом фрагменте показан переход выполнения программы на оператор с меткой 3.

2) Арифметический оператор условной передачи управления

IF (W) N1, N2, N3

Здесь W — арифметическое выражение, N1, N2, N3 — метки операторов (могут совпадать). Этот оператор изменяет последовательный ход выполнения программы, но управление может быть передано в три места программы в зависимости от знака арифметического выражения.

Если $W<0$, то передача управления на оператор с меткой N1.
Если $W=0$, то передача управления на оператор с меткой N2.
Если $W>0$, то передача управления на оператор с меткой N3.

Арифметический оператор IF является устаревшей конструкцией фортрана, более употребителен логический оператор IF, описываемый ниже.

Кроме того, применение трех различных меток в этом операторе нарушает принятое в гл. 2 соглашение об использовании только логической схемы с двойным ветвлением. Однако противоречие снимается, если всегда в арифметическом операторе IF употреблять две совпадающие метки, что мы и будем делать.

Пример.

```
      D=B**2-4.*A*C
      IF (D) 2, 1, 1
      ...
```

```
2       ALPHA=3.
        ...
1       ALPHA=26.
        ...
```

С помощью операторов передачи управления GO TO и IF или IF можно организовать в программе цикл.

Цикл — это совокупность операторов, которые выполняются в программе несколько раз.

Пример. Найти N!=1*2*...*N, N задано.

Обозначим NF результат

```
        NF=1
        I=1
1       NF=NF*I
2       IF (I-N) 3, 5, 5
3       I=I+1
4       GO TO 1
5        ...
```

Операторы 1—4 — цикл. Выход из цикла дает оператор с меткой 2 при I=N.

Цикл можно организовать с помощью специального оператора цикла.

3) Оператор цикла

```
DO N I=M1, M2, M3
```

Здесь N — метка последнего выполняемого оператора цикла, I — целая переменная, M1, M2, M3 — целые константы и выражения

Первый оператор цикла — DO, последний помечен меткой N. Цикл повторяется при различных I. Начальное значение I равно M1, конечное — M2, шаг изменения I равен M3. Если M3=1, то можно записать оператор цикла короче:

```
DO N I=M1, M2
```

Значение шага M3 не должно быть равным 0, но может быть отрицательным.

Пример. Предыдущая задача вычисления N! с помощью оператора цикла решается так:

```
        NF=1
        DO 5 I=1, N
5       NF=NF*I
```

Последний оператор цикла не должен быть оператором передачи управления, так как сам оператор DO управляет передачей управления на начало цикла. Иногда это правило не дает возможностей выйти из цикла. Тогда последним оператором цикла ставят оператор продолжения

```
N       CONTINUE
```

где N — метка, цикл записывают так:

```
      DO  N  I=M1,M2,M3
      ...
N     CONTINUE
```

В области действия одного оператора DO может содержаться другой оператор DO со своей областью действия — вложенные циклы, например

```
      DO  N  I=M1,M2,M3
      ...
      DO  K  J=N1,N2,N3
      ...
K     CONTINUE
      ...
N     CONTINUE
```

Проиллюстрируем действие оператора DO на двух примерах.

1. *Вычисление конечной суммы числовой последовательности.* Пусть необходимо найти сумму S:

$$S=\sum_{n=1}^{30}\exp(-n).$$

Программа решения этой задачи может иметь вид

```
      REAL S,X
      INTEGER N
      N=30
C     ПЕРЕД ЦИКЛОМ СУММИРОВАНИЯ ПЕРЕМЕННОЙ,
C     В КОТОРОЙ БУДЕТ ПРОИЗВОДИТЬСЯ НАКОПЛЕНИЕ
C     СУММЫ, ПРИСВАИВАЕТСЯ НУЛЕВОЕ ЗНАЧЕНИЕ
      S=0.
      DO 1 I=1,N
      X=FLOAT(I)
1     S=S+EXP(-X)
C     ВЫВОД НА ТЕРМИНАЛ ЗНАЧЕНИЯ СУММЫ
      WRITE (5,2) S
2     FORMAT (2X,'S=',E13.6)
      END
```

2. *Вычисление произведения W двух матриц U и V.* Формула произведения матриц

$$w_{i,j}=\sum_{k=1}^{m}u_{i,k}\,y_{k,j},\quad 1\leqslant i\leqslant l,\quad 1\leqslant j\leqslant n.$$

Пусть известны значения $m=l=n=3$. Программа вычисления элементов матрицы w может иметь вид

```
      REAL U(3,3),Y(3,3),W(3,3)
C     ВВОД МАТРИЦ U,Y
      READ (5,1) U,Y
1     FORMAT (3F4.1)
```

```
C       ВЫЧИСЛЕНИЕ W(I,J)
        DO 10 I=1,3
        DO 10 J=1,3
        W(I,J)=0.
        DO 10 K=1,3
10      W(I,J)=W(I,J)+U(I,K)*Y(K,J)
C       ВЫВОД НА ТЕРМИНАЛ МАТРИЦЫ W
        WRITE (5,100) W
100     FORMAT (2X,E13.6)
        END
```

Для ввода — вывода элементов массива применяется конструкция фортрана, которая называется неявным циклом. Эта конструкция аналогична действию оператора цикла DO.

4) Неявный цикл записывается в форме

(A(I), I=M1, M2, M3)

для одномерного массива A,

((A(I,J), I=M1,M2,M3), J=N1,N2,N3)

для двумерного массива A, где M1, M2, M3, N1, N2, N3 имеют тот же смысл, что и в определении цикла.

Ввод (вывод) с помощью неявного цикла программируется, например, следующим образом:

```
        READ (5,1) (W(I), I=1,3)
1       FORMAT (F4.1)
```

неявный цикл удобен для ввода элементов массива с переменными размерами. Например,

```
        REAL A (10,5)
C       ВВОД ЧИСЛА СТРОК И СТОЛБЦОВ МАТРИЦЫ A(I,J)
        READ (5,1) M,N
1       FORMAT (2I2)
C       ВВОД ЭЛЕМЕНТОВ МАТРИЦЫ A(I,J)
        READ (5,2) ((A(I,J), I=1,M),J=1,N)
2       FORMAT (F4.1)
```

3.2.9. Логические выражения, условный логический оператор. Логические выражения состоят из следующих компонентов: выражений отношения, логических операций, логических переменных и констант.

Выражения отношения имеют вид

W1 □ W2

где W1, W2 — арифметические выражения, □ — знак операции отношения — один из следующих:

.LT. меньше $<$
.LE. меньше или равно $\leqslant$
.EQ. равно $=$
.NE. не равно $\neq$
.GT. больше $>$
.GE. больше или равно $\geqslant$

Примеры.

1. $3>4$. Запись на фортране **3.GT.4.**

2. $(A+B)<(G+D)$.. Запись на фортране (A+B).LT.(C+D).

Значением выражения отношения может быть либо «истина» — .TRUE., либо «ложь» .FALSE..

В примере 1—.FALSE., в примере 2 — зависит от входящих параметров.

Логическая переменная — величина, принимающая в процессе вычислений два значения: .TRUE. либо FALSE.

Логические выражения имеют вид

$$Z1\Delta Z2\Delta Z3...$$

Здесь Z1, Z2, Z3, ... — логические элементы: переменные, константы, выражения отношения, Δ — знак логической операции.

Логические операции

Знак операции Δ	*Пример*	*Значение операции*
.AND.	A.AND.B	Выражение истинно, когда A и B истинны
.OR.	A.OR.B	Выражение истинно, когда истинно либо A, либо B, либо они оба
.NOT.	.NOT.A	Выражение истинно, когда ложно A

Примеры логических выражений:

1.A.AND.B.OR.C
2.X.LT.Y.AND.Y.LT.Z

Логическое выражение в примере 2 истинно, если переменная Y принадлежит интервалу $X<Y<Z$.

Логическое выражение

A.GT.1.0.AND..NOT.C.EQ.D.OR.B.LT.C

иллюстрирует две особенности. Во-первых, две логические операции не могут стоять в выражении рядом, кроме случая, когда вторая операция .NOT.. Во-вторых, в логическом выражении имеется неоднозначность. Приведенное выражение может означать

A.GT.1.0.AND..NOT.(C.EQ.D.OR.B.LT.C)

а может

(A.GT.1.0.AND..NOT.C.EQ.D).OR.B.LT.C

Такую неоднозначность можно устранить применяя скобки, так же как и в арифметических выражениях. Когда неоднозначность не устраняется скобками, выражения отношения и логические операции выполняются в порядке старшинства. В порядке убывания старшинства (вместе с арифметическими операциями) эта последовательность имеет вид

возведение в степень
умножение, деление
сложение, вычитание
выражения отношения
.NOT.
.AND.
.OR.

С учетом этой иерархии приведенное выше логическое выражение означает второй вариант расстановки скобок

Логический оператор присваивания имеет вид

Q=R,

где Q — логическая переменная, R — логическое выражение.

Условный логический оператор IF. Вид этого оператора

IF (R) S

Здесь R — логическое выражение, S — выполняемый оператор, кроме двух операторов: DO и IF. Если R истинно, то выполняется оператор S, если R ложно, то выполняется следующий за IF оператор.

Примеры.

1. Вычисление функции в точке x

$$Y=F(X)=\begin{cases} -x+2{,}1, & x<-2, \\ 1+2x^3, & -2\leqslant x\leqslant 2, \\ x+3{,}5, & x>2 \end{cases}$$

с помощью логического IF запрограммировано в следующем фрагменте:

```
      IF (X.LT.(-2)) GO TO 2
      IF (X.LE.2) GO TO 1
      Y=X+3.5
      GO TO 3
1     Y=1.+2.*X**3
      GO TO 3
2     Y=(-X)+2.1
3     CONTINUE
```

2. Выход на печать при достижении заданной точности и засылка X1 в X0, если точность не достигнута. Текущая точность E1, заданная E.

```
      IF (E1.LE.E) GO TO 10
      X0=X1
      ...
10    WRITE (6,11)X1
11    FORMAT (E13.6)
```

3. Определение максимального элемента матрицы $W(I, J)$, $1 \leqslant I, J \leqslant 10$:

```
      AMAX=W(1,1)
      DO 5 I=1,10
      DO 5 J=1,10
      IF (W(I,J).GT.AMAX) AMAX=W(I,J)
5     CONTINUE
```

3.2.10. Функция-формула, функция-подпрограмма, подпрограмма. Если в программе необходимо несколько раз вычислять значение функции, которой нет среди библиотечных, и эта функция может быть определена одним арифметическим выражением, то можно использовать конструкцию фортрана, которая называется функция-формула.

1. *Функция-формула* определяется арифметическим выражением в начале программы до первого выполняемого оператора

```
F(P1,P2,...,PK)=W
```

где F — индентификатор (имя функции), P1,...,PK — идентификаторы формальных аргументов, W — арифметическое выражение — последовательность операций над аргументами.

После определения функцией-формулой пользуются так же, как и библиотечной функцией.

Аргументы в обращении к функции называются фактическими параметрами. Количество, последовательность и тип формальных и фактических параметров должны совпадать. Фактические параметры могут быть константами, переменными, функциями, арифметическими выражениями.

Тип данных функции-формулы определяется явным образом в операторе описания типа (либо неявным образом по первой букве идентификаторов).

Примеры.

```
Определения: ANORMA(X,Y,Z)=SQRT(X**2+Y**2+Z**2)
             F(X)=-1.+X**2+X+0.01*SIN(X)
Обращения:   Y=ANORMA(2.,COS(B),0.3)+1.
             Z=F(X0)
```

При появлении в программе имени функции-формулы выполняется подстановка фактических параметров на место соответствующих им формальных в определении функции-формулы. После этого вычисляется арифметическое выражение, определяющее функцию, а затем результат используется для вычисления выражения, содержащего обращение к функции.

Перед обращением к функции-формуле всем фактическим параметрам должны быть уже присвоены значения.

Обращение к функции-формуле может быть только в той программной единице (это понятие вводится в этом разделе ниже), где она определена.

2. *Функция-подпрограмма* применяется тогда, когда вычисление значения функции не сводится к одному арифметическому выражению. Определяется функция-подпрограмма следующим образом:

```
[ТИП] FUNCTION F(P1,P2,...,PK)
...
RETURN
...
END
```

где ТИП — указатель типа данных результатов (значения функции); знаки [...] означают, что этот указатель может отсутствовать, F — имя функции, P1,..., PK — идентификаторы формальных параметров, RETURN — оператор возврата в вызывающую программу, END — оператор конца самостоятельной программной единицы.

Пример.

Функция-подпрограмма, выбирающая меньшее из двух чисел:

```
      REAL FUNCTION SMALL(A,B)
      REAL A,B
      IF (A.LT.B) GO TO 1
      SMALL=B
      GO TO 2
1     SMALL=A
2     RETURN
      END
```

Обращение к этой функции в вызывающей программе может быть, например, таким:

```
R=2.+SMALL(X,Y)
```

В этот момент вызывающая программа передает управление функции-подпрограмме SMALL, происходит замена формальных параметров A, B на фактические X, Y (им уже должны быть присвоены значения). После этого выполняются операторы функции-подпрограммы. Результат вычислений присваивается имени функции, в рассмотренном примере: SMALL. Затем управление возвращается в вызывающую программную единицу и происходит вычисление R.

Программа на фортране может состоять из главной программы и множества FUNCTION-подпрограмм, каждая из которых заключена между заголовком FUNCTION и оператором END. Это самостоятельные программные единицы. В отличие от функции — формулы FUNCTION не определяется в другой программной единице, а вызывается из нее.

При трансляции функция-подпрограмма рассматривается отдельной программой, т. е. ее можно транслировать без главной программы, вызывающих ее и вызываемых ею программ. Это дает следующие возможности в программировании. Большую программу, разбивая на отдельные небольшие подпрограммы, можно отлаживать на ЭВМ отдельно, а затем собирать в единую программу.

Используемые в функции-подпрограмме переменные являются локальными, если они не объявлены в операторе COMMON (см. ниже). Поэтому переменная A в одной подпрограмме совершенно отлична от переменной A в другой программе (занимают различные ячейки памяти). Локальным переменным можно присвоить значения только в той программной единице, где они определены, если они не передаются как фактические параметры в другую программную единицу.

Как и в функции-формуле, должно соблюдаться соответствие фактических и формальных параметров по количеству, последовательности, типу.

В качестве примера рассмотрим программу вычисления полинома

$$2x^4-3x^2+x-1$$

в точке $x=0{,}3$ по схеме Горнера. Схему Горнера реализуем в виде функции-подпрограммы. В главной программе осуществим ввод, вывод числовой информации и вызов функции-подпрограммы. Напомним, что вычисление значения $P_n(x)$

$$P_n(x)=a_0x^n+a_1x^{n-1}+\ldots+a_{n-1}x+a_n$$

в точке x по схеме Горнера выполняется по формуле

$$P_n(x)=(\ldots(((a_0x+a_1)x+a_2)x+a_3)x+\ldots+a_{n-1})x+a_n.$$

```
C       ГЛАВНАЯ ПРОГРАММА
        REAL X,A(5),Y
        READ (5,1) X,A
1       FORMAT (6F4.1)
        Y=P(5,A,X)
        WRITE (5,2) Y
2       FORMAT (2X,'Y=',E13.6)
        END
C       ФУНКЦИЯ-ПОДПРОГРАММА
        FUNCTION P(M,A,X)
        REAL A(M),X
        INTEGER M
        P=A(1)
        DO 1 J=2,M
1       P=P*X+A(J)
        RETURN
        END
```

В соответствии с числовыми данными задачи следует, согласно заказанному формату, набрать на клавиатуре терминала числа.

3. *Подпрограмма* обладает более широкими возможностями, нежели функция-подпрограмма. В качестве результата подпрограмма может передавать в вызывающую программу либо несколько числовых значений, либо ни одного.

Подпрограмма оформляется следующим образом:

```
      SUBROUTINE S(P1,P2,...,PK)
      ...
      ...
      RETURN
      ...
      END
```

Первый оператор подпрограммы — SUBROUTINE, S — имя подпрограммы, P1, P2, ..., PK — формальные параметры. Формальные параметры, являющиеся идентификаторами массивов, описываются внутри подпрограммы. Последний оператор END. Возвращение к вызывающей программе осуществляется через оператор RETURN.

Обращение к подпрограмме (вызов) производится в вызывающей программе посредством оператора CALL в форме

```
      CALL S(Q1,...,QK)
```

Здесь S — имя вызываемой подпрограммы, Q1, Q2,...,QK — фактические параметры.

Пример. Подпрограмма транспонирования квадратной матрицы

```
      SUBROUTINE TRANSP(N,A,B)
      REAL A(N,N),B(N,N)
      DO 10 I=1,N
      DO 10 J=1,N
10    B(I,J)=A(J,I)
      RETURN
      END
```

Транспонированная матрица располагается в массиве B. В вызывающей программе необходимо иметь следующие операторы:

```
      REAL Q(8,8),R(8,8)
      ...
      CALL TRANSP(8,Q,R)
```

После работы подпрограммы TRANSP в двумерном массиве R размером 8×8 будет расположена транспонированная матрица Q. Элементам массива Q должны быть присвоены числовые значения до вызова TRANSP.

Список формальных параметров подпрограммы может быть пуст. Например,

```
      SUBROUTINE ER
      WRITE (5,1)
1     FORMAT ('ERROR')
      RETURN
      END
```

В таком случае при вызове подпрограммы указывается только имя

```
      CALL ER
```

Имени подпрограммы в отличие от функции-подпрограммы не присваивается значение. Значения переменных передаются в подпрограмму через входные параметры (при замене формальных параметров на фактические) и возвращаются в вызывающую программу через выходные параметры. Один и тот же параметр может быть и входным и выходным.

В приведенном выше примере

входные параметры: N,A

выходные параметры: B

Соответствие фактических и формальных параметров по количеству, порядку и типу — взаимно однозначное как в функции-формуле, так и в функции-подпрограмме. Заметим, что аргументы функций и подпрограмм не подвергаются неявным преобразованиям, поэтому несоответствие типа параметров является грубой ошибкой. Транслятор этот факт не отмечает в диагностике ошибок, поэтому на этапе вычислений, как правило, происходит аварийный останов.

Функция-подпрограмма и подпрограмма называются процедурами фортрана.

Поясним общую структуру произвольной программы. Программы могут иметь простую и модульную структуру. Программа простой структуры представляет собой последовательность невыполняемых и выполняемых операторов, из которых хотя бы один должен быть выполняемый, последний оператор END. Имеем

```
S1
S2
...
SN
```

Оператор SN есть END. Программа модульной структуры состоит из нескольких самостоятельных единиц (подпрограмм, функций-подпрограмм), одна из которых — главная программа. Главная программа может обращаться к подпрограммам, эти подпрограммы — к другим и т. д. Выполнение программы начинается первым выполняемым оператором главной программы и заканчивается в главной программе.

Следует отметить, что предпочтительнее писать программы модульной структуры. Накапливая подпрограммы, пользователь

в дальнейшем, как из кирпичиков, строит здание программы в узкой, специальной области, все меньше и меньше затрачивая усилий на программирование. Таким образом, мы приходим к организации личной библиотеки программ пользователя, а также к использованию уже готовых библиотек и пакетов программ.

Описание библиотеки фортран-программ по основным разделам вычислительной математики приведено в гл. 12.

3.2.11. Невыполняемые операторы. Как уже отмечалось, программа состоит из последовательности невыполняемых и выполняемых операторов.

Выполняемые операторы приводят к определенным действиям, невыполняемые служат для описания величин, массивов, формата ввода и вывода, распределения памяти. Выполняемые операторы — это операторы присваивания, управления, ввода, вывода.

Невыполняемые операторы:

1) Операторы описания типа (INTEGER, REAL, DOUBLE PRECISION, LOGICAL, COMPLEX)
2) EXTERNAL
3) DIMENSION
4) COMMON
5) EQUIVALENCE
6) Оператор начальных данных DATA
7) Оператор FORMAT
8) Оператор определения функции FUNCTION
9) Оператор определения подпрограммы SUBROUTINE

Порядок следования в программе невыполняемых операторов с 1) по 6), указанный выше, желателен при программировании. Операторы с 1) по 6) должны быть расположены до первого выполняемого оператора.

Некоторые из невыполняемых операторов уже были определены. В этом пункте описываются четыре невыполняемых оператора:

COMMON, EQUIVALENCE, EXTERNAL, DATA

Как уже отмечалось, переменные в самостоятельных программных единицах (главная программа, функция-подпрограмма, подпрограмма) имеют локальное определение.

Чтобы переменные различных программных единиц имели глобальное определение во всей программе, употребляется оператор COMMON (общий).

Иногда говорят, что через оператор COMMON передают значения переменных из одной программной единицы в другую. Один из способов передачи — через список входных и выходных параметров — рассмотрен в предыдущем пункте.

Оператор имеет следующую форму записи:

COMMON P1,P2,...,PK

Здесь P1, P2, ..., PK — наименование глобальных переменных. Этот оператор должен стоять в каждой подпрограмме и главной программе до первого выполняемого оператора. Размерность массивов можно указать в списке переменных.

Пример.

В вызывающей программе

```
COMMON  A,B,C(10),Q(15,15)
...
END
```

В подпрограмме или функции-подпрограмме

```
COMMON  A,B,C(10),Q(15,15)
...
END
```

Если размерность массива описана в операторе описания типа или DIMENSION, то в операторе COMMON этого делать не нужно.

Например, описание

```
REAL  W(9),T(14)
COMMON  W,T
```

и описание

```
COMMON  W(9),T(14)
```

эквивалентны.

Программа может содержать любое количество операторов COMMON. Но в этом случае каждому блоку глобальных переменных, кроме, возможно, одного, следует присвоить имя — идентификатор — и заключить между двумя наклонными чертами.

Пример.

Оператор

```
COMMON/STAR1/A,B,C(10)/STAR2/Q(5,5)
```

в одной программной единице и оператор

```
COMMON/STAR1/T,R,Z(10)/STAR2/D(5,5)
```

в другой определяют одни и те же переменные, так как транслятор отводит для них одни и те же ячейки памяти. Нужно следить только, чтобы тип, размерность и длина соответствующих переменных совпадали.

Один из возможных способов экономии памяти — это использование одного и того же места памяти для хранения разных величин, участвующих в выполнении данной программной единицы не одновременно. Этой цели служит оператор EQUIVALENCE.

Формат оператора:

```
EQUIVALENCE (список),..., (список)
```

В скобках (список) идентификаторы разделены запятыми. Например,

```
EQUIVALENCE (A,B,C),(L,M)
```

Элементы A, B, C будут запоминаться в одном месте памяти (32 разряда), элементы L, M также займут только одну ячейку (16 разрядов).

Чтобы установить эквивалентность массивов одинаковой размерности, нужно установить эквивалентность, например, их первых компонент:

```
REAL X(10),Y(10)
EQUIVALENCE (X(1),Y(1))
```

Если в функции-подпрограмме или подпрограмме в качестве фактического параметра используется идентификатор (имя) другой функции-подпрограммы или подпрограммы, то в вызывающей программе этот идентификатор должен быть описан с помощью оператора EXTERNAL (внешний). Вид оператора

```
EXTERNAL F1,F2,...,FN
```

где F1, F2, ..., FN — наименование функций, подпрограмм, которые в данной программе могут передаваться в качестве фактических параметров другим функциям и подпрограммам.

Пример.

Функция-подпрограмма

```
FUNCTION R(X,Y,Z)
R=X*Y*Z(X)
RETURN
END
```

Функция-подпрограмма

```
FUNCTION ST(X)
ST=X+SQRT(X**2)
RETURN
END
```

Вызывающая программа

```
EXTERNAL ST
...
X=2.
C=D+R(X,X,ST)
```

Если в операторе EXTERNAL стоит имя библиотечной функции фортрана, то оно становится именем функции, написанной пользователем, и подавляет вызов библиотечной.

Например, пользователь написал, по его мнению, более эффективную программу вычисления $\sin(x)$. Тогда в главной программе следует указать

```
EXTERNAL SIN
```

и присоединить программу, реализующую эти вычисления:

```
FUNCTION SIN(X)
...
SIN=...
...
```

```
      RETURN
      END
```

Тогда при обращении в головной программе

```
      Y=SIN(X)
```

будет вызываться программа пользователя, а не библиотечная.

Ранее было показано, что числовые значения переменным можно присвоить с помощью оператора присваивания. Однако если переменных много, то удобнее начальные значения присвоить до выполнения программы с помощью оператора DATA, который имеет следующий вид:

```
DATA P1,...,PN/C1,...,CN/,Q1,...,QM/Z1,...,ZM/
```

Здесь P1,...,PN — имена переменных; C1,...,CN — соответствующие константы; Q1,...,QM — переменные; Z1,...,ZM — соответствующие константы и т. д. Перед константами можно ставить множитель N, указывающий число повторений этой константы в списке констант.

Пример.

```
DATA X,Y,Z/3*2.05/,M(1,2)/3/
```

Этот оператор эквивалентен следующим операторам присваивания:

```
      X=2.05
      Y=2.05
      Z=2.05
      M(1,2)=3
```

Место оператора DATA в программной единице — после всех невыполняемых операторов, за исключением, возможно, оператора FORMAT, который может находиться в любом месте программы.

Все переменные, используемые в программе, перед вычислениями должны быть каким-либо способом инициализированы, т. е. им следует присвоить значения (операторы присваивания, ввода, DATA). Плохим «стилем» программирования является отказ от инициализации и предположение, что ячейки памяти перед входом в программу будут очищены (хотя на некоторых ЭВМ так и происходит).

●3.3. Программирование элементарных вычислительных алгоритмов

3.3.1. Введение. В настоящем пункте рассматриваются приемы программирования на фортране элементарных вычислительных алгоритмов. Элементарными эти алгоритмы называются потому, что дальнейшее разбиение их на более мелкие вычислительные процедуры практически нецелесообразно. Логика таких алгоритмов сравнительно проста.

Рассмотрение ведется в соответствии с разделами библиотеки фортран-программ (см. гл. 12). Программы оформляются в виде подпрограмм или функций-подпрограмм с именами, согласованными с библиотечной системой имен. Например, имя программ A4S1, B5S2 означает, что они относятся к разделам A4 «Интегрирование» или B5 «Нелинейная оптимизация», S1, S2 — «простая» программа номер 1 или 2.

Естественно, что «простые» программы для элементарных алгоритмов могут быть также включены в библиотеку программ и использоваться наряду со «сложными» программами. Отличие элементарных вычислительных алгоритмов связано в основном со значительно меньшей универсальностью в сравнении с библиотечными алгоритмами. Их область применения более узкая; как правило, нет возможности автоматического контроля погрешности вычислений, нет диагностики возможной ошибки и т. п. Однако при всех недостатках элементарные программы имеют и преимущества, которые могут «перевесить» недостатки: 1) эти элементарные программы малы по числу операторов, т. е. занимают меньше ячеек памяти ЭВМ, чем универсальные; 2) элементарные программы пишет пользователь для своей задачи, а никто лучше него самого не знает алгоритм и данные.

Элементарные или «простые» алгоритмы и программы не означают, что они плохи. Их эффективность может быть значительно выше универсальных алгоритмов, но для специальных классов задач.

Наконец, имея опыт программирования элементарных алгоритмов, можно, следуя идее структурной алгоритмизации (см. гл. 2), проектировать достаточно сложные алгоритмы и программы из элементарных.

3.3.2. Табулирование функций. Составление таблиц функций называют табулированием. Предварительно вычисленные таблицы, хранимые в памяти ЭВМ, могут затем значительно сократить дальнейшие вычисления.

Пусть необходимо иметь таблицу значений функции $f(x)$ с заданным постоянным шагом h в n точках x_0+ih, $0\leqslant i\leqslant n-1$, начальное значение x_0 задается.

Входные данные задачи: $f(x)$, h, x_0, n;
выходные данные: $f(x_0), f(x_1), \ldots, f(x_{n-1})$.

Подпрограмме присвоим имя Z0S0. Поскольку в библиотеке нет раздела табулирования, введен новый индекс Z0.

Для вычисления выходных данных — массива $y_i=f(x_i)$ — используется следующий фрагмент:

```
      X=X0
      DO 1 I=1,N
      Y(I)=F(X)
1     X=X+H
```

Здесь используется F — имя функции-подпрограммы, которая должна вычислять $f(x)$. Теперь, следуя правилам фортрана, оформим подпрограмму Z0S0. Имеем

```
      SUBROUTINE Z0S0(F,H,X0,N,Y)
      REAL H,X0,Y(N),X
      INTEGER N
      X=X0
      DO 1 I=1,N
      Y(I)=F(X)
1     X=X+H
      RETURN
      END
```

Пусть требуется составить таблицу функции

$$f(x)=e^{-x}+\operatorname{arctg} x$$

в интервале $[0, 1]$ с шагом $h=0,01$. Программа может иметь следующий вид:

```
      REAL X0,H,Y(100)
      INTEGER N
      EXTERNAL F
      DATA X0,H,N/0.,0.01,100/
      CALL Z0S0(F,H,X0,N,Y)
      END
C     ВНЕШНЯЯ ФУНКЦИЯ-ПОДПРОГРАММА, ВЫЧИС-
C     ЛЯЮЩАЯ F(X)
      FUNCTION F(X)
      REAL X
      F=EXP(-X)+ATAN(X)
      RETURN
      END
```

Пусть необходимо иметь таблицу значений функции $f(x)$ в точках x_i, $1 \leqslant i \leqslant n$, шаг $h_i = x_{i+1} - x_i$, вообще говоря, переменный. В этом случае:

входные данные задачи: x_i, $1 \leqslant i \leqslant n$, $f(x)$;

выходные данные: $f(x_1), f(x_2), \ldots, f(x_n)$.

Подпрограмме табулирования с переменным шагом присвоим имя Z0S1:

```
      SUBROUTINE Z0S1(F,X,N,Y)
      REAL X(N),Y(N)
      INTEGER N
      DO 1 I=1,N
1     Y(I)=F(X(I))
      RETURN
      END
```

3.3.3. Аппроксимация функций. Простейшим способом приближения функции $f(x)$, заданной в n точках x_i, $1 \leqslant i \leqslant n$, является линейная интерполяция по двум точкам.

Для любого x, $x_1 \leqslant x \leqslant x_n$, значение $f(x)$ приближенно заменяется формулой

$$f(x) \simeq p(x) = f(x_i)\frac{x_{i+1}-x}{x_{i+1}-x_i} + f(x_{i+1})\frac{x-x_i}{x_{i+1}-x_i}, \quad x_i \leqslant x \leqslant x_{i+1},$$

описывающей прямую, которая проходит по двум точкам $(x_i, f(x_i)$ и $(x_{i+1}, f(x_{i+1}))$ плоскости (x, y).

Напишем программу линейной интерполяции в двух вариантах: 1) когда $f(x)$ задана таблично с постоянным шагом; 2) с переменным.

Вариант 1.

Входные данные: $f(x_i)$, $1 \leqslant i \leqslant n$, x_0, h, x;

выходные данные: $p(x)$.

Основным моментом алгоритма линейной интерполяции является поиск интервала $[x_i, x_{i+1}]$, которому принадлежит точка x, где производится вычисление $p(x)$. Чтобы найти номер i, следует вычислить целую часть дроби $[(x-x_1)/h]$. Среди библиотечных функций фортрана есть соответствующая функция INT(X), реализующая выделение целой части. Поэтому программа может иметь следующий вид:

```
      FUNCTION A3S0(Y,N,X1,H,X)
      REAL Y(N),X1,H,P,XM,XM1
      INTEGER N,M
C     ВХОДНЫЕ ПАРАМЕТРЫ
C     Y(N) — МАССИВ ЗНАЧЕНИЙ ФУНКЦИИ
C        N — РАЗМЕРНОСТЬ МАССИВА
C        H — ШАГ ТАБЛИЦЫ
C        X — КООРДИНАТА, В КОТОРОЙ ПРОИЗВОДИТСЯ
C     ИНТЕРПОЛЯЦИЯ
C     ВЫХОДНЫЕ ПАРАМЕТРЫ
C     A3S0 — ПРИБЛИЖЕННОЕ ЗНАЧЕНИЕ ФУНКЦИИ В
C     ТОЧКЕ X
      M=INT((X-X1)/H)
      XM=X1+M*H
      XM1=XM+H
      P=(Y(M)*(XM1-X)+Y(M+1)*(X-XM))/H
      A3S0=P
      RETURN
      END
```

Вариант 2.

Входные данные: $x_i, f(x_i)$, $1 \leqslant i \leqslant n$, z;

выходные данные: $p(z)$. Будем предполагать, что $z \neq x_n$. В этом варианте поиск интервала $[x_i, x_{i+1}]$, содержащего точку z, можно провести различными способами. Воспользуемся простым перебором, а именно: в цикле по i, $1 \leqslant i \leqslant n$, используем логический оператор IF. Когда выражение в скобках станет истинным, цикл прекратим. Имеем соответствующий фрагмент

```
      I=1
1     IF (Z.GE.X(I).AND.Z.LT.X(I+1)) GO TO 2
      I=I+1
      GO TO 1
2     ...
```

Теперь полная программа может быть полностью представлена следующим образом:

```
      FUNCTION A3S1(X,Y,N,Z)
      REAL X(N),Y(N),P,Z
      INTEGER N,I
      I=1
1     IF (Z.GE.X(I).AND.Z.LT.X(I+1)) GO TO 2
      I=I+1
      GO TO 1
2     P=(Y(I)*(X(I+1)-Z)+Y(I+1)*(Z-X(I)))/(X(I+1)-X(I))
      A3S1=P
      RETURN
      END
```

3.3.4. Интегрирование. В качестве элементарного алгоритма численного интегрирования примем формулу Симпсона

$$Q=\frac{h}{3}\left(f_0+4\sum_{i=1}^{m}f_{2i-1}+2\sum_{i=1}^{m-1}f_{2i}+f_{2m}\right).$$

Здесь h — шаг по оси x, $f_i=f(x_i)$, $0\leqslant i\leqslant 2m$, $x_{i+1}-x_i=h$ (см. рис. 7.7). Для написания программы вычисления Q по значениям функции f_i заметим, что она может быть представлена в виде

$$Q=\frac{h}{3}\left(\sum_{i=0}^{m-1}(f_{2i}+4f_{2i+1}+f_{2i+2})\right).$$

Каждое слагаемое в сумме имеет вид

$$f(x)+4f(x+h)+f(x+2h),$$

где $x=x_{2i}$, $x_0=a$. Поэтому для вычисления суммы можно использовать следующий фрагмент:

```
      Q=0.
      X=A
      DO 1 I=2,N,2
      Q=Q+F(X)+4.*F(X+H)+F(X+2.*H)
1     X=X+2.*H
```

Здесь $N=2m$, $H=h$, $A=a$. Теперь уже не представляет труда оформить соответствующую программу:

```
      FUNCTION A4S0(A,B,N,F)
      REAL A,B,H,Q
      INTEGER N
C     ВХОДНЫЕ ПАРАМЕТРЫ
```

```
C       A — НИЖНИЙ ПРЕДЕЛ ИНТЕГРИРОВАНИЯ
C       B — ВЕРХНИЙ ПРЕДЕЛ ИНТЕГРИРОВАНИЯ
C       N — ЧЕТНОЕ ЧИСЛО ОТРЕЗКОВ ИНТЕГРИРОВАНИЯ
C       F — ИМЯ ВНЕШНЕЙ ФУНКЦИИ ВИДА
C       REAL FUNCTION F(X)
C       REAL X
C       ВЫЧИСЛЯЮЩЕЙ ЗНАЧЕНИЕ ПОДИНТЕГРАЛЬНОЙ
C       ФУНКЦИИ
C       A4S0 — ПРИБЛИЖЕННОЕ ЗНАЧЕНИЕ ИНТЕГРАЛА
        H=(B-A)/N
        Q=0.
        X=A
        DO 1 I=2,N,2
        Q=Q+F(X)+4.*F(X+H)+F(X+2.*H)
1       X=X+2.*H
        Q=Q*H/3
        A4S0=Q
        RETURN
        END
```

В гл. 7 показано, что оценка погрешности приближенного значения интеграла может быть получена вычислением Q_1 с шагом H, затем Q_2 с шагом H/2, а абсолютная погрешность ε определяется выражением (по правилу Рунге)

$$\varepsilon = |Q_1 - Q_2|/15.$$

Вычислим

$$\int_0^2 \exp(\sin x)\,dx$$

с числом отрезков разбиения интервала [0, 2], равным N=100, 200. Результаты (приближенное значение интеграла Q_2 и ошибку ε) будем выдавать на терминал:

```
        REAL A,B,E,Q1,Q2
        INTEGER N
        EXTERNAL F
        DATA A,B,N/0.,2.,100/
        Q1=A4S0(A,B,N,F)
        Q2=A4S0(A,B,2*N,F)
        E=ABS((Q1-Q2)/15.)
        WRITE (5,1) Q2,E
1       FORMAT (2X,E13.6,2X,E13.6)
        END
C       ВНЕШНЯЯ ФУНКЦИЯ F(X)
        FUNCTION F(X)
        F=EXP(SIN(X))
        RETURN
        END
```

3.3.5. Суммирование рядов. Суммирование конечной числовой последовательности

$$S=\sum_{i=1}^{n} a_i$$

выполняется с помощью оператора цикла DO тремя операторами:

```
      S=0.
      DO 1 I=1,N
1     S=S+A(I)
```

где A — имя функции-подпрограммы, вычисляющей a_i. Например,

```
      REAL FUNCTION A(I)
      INTEGER I
      ...
      RETURN
      END
```

Суммирование бесконечных числовых рядов

$$S=\sum_{i=1}^{\infty} a_i$$

выполняется обычно с помощью цикла, организованного логическим оператором IF и оператором безусловной передачи управления GO TO. Выход из цикла осуществляется при достижении значения вычисляемой точности ε_1 заданной ε.

Рассмотрим программу суммирования сходящегося знакопеременного ряда, для которого легко вычислить оценку погрешности; она меньше модуля первого отброшенного члена ряда

$$|S-S_N|=\left|S-\sum_{i=1}^{N} a_i\right|\leqslant|a_{N+1}|.$$

Приведем соответствующую программу:

```
      FUNCTION A5S0(E,A)
      REAL E,S
      INTEGER N
C     ВХОДНЫЕ ПАРАМЕТРЫ
C     E — ЗАДАННАЯ АБСОЛЮТНАЯ ПОГРЕШНОСТЬ
C     A — ИМЯ ВНЕШНЕЙ ФУНКЦИИ-ПОДПРОГРАММЫ,
C         ВЫЧИСЛЯЮЩЕЙ ОБЩИЙ ЧЛЕН ЗНАКОПЕРЕ-
C         МЕННОГО РЯДА
C     ВЫХОДНЫЕ ПАРАМЕТРЫ
C     A5S0 — ПРИБЛИЖЕННОЕ ЗНАЧЕНИЕ СУММЫ РЯДА
      DATA N,S/1,0./
1     S=S+A(N)
      IF (ABS(A(N)).GT.E) GO TO 2
      GO TO 3
2     N=N+1
```

```
      GO TO 1
3     A5S0=S
      RETURN
      END
```

В качестве иллюстрации применения этой программы рассмотрим суммирование ряда

$$S=\sum_{i=1}^{\infty}(-1)^2(i^2+1)^{-1}$$

с точностью $\varepsilon=10^{-4}$. Программа может иметь вид

```
      REAL E,S
      EXTERNAL A
      DATA E/1.E-4/
      S=A5S0(E,A)
      WRITE (5,1) S
1     FORMAT (2X,'S=',E11.4)
      END
C     ...
      FUNCTION A(I)
      INTEGER I
      IF (MOD(I,2).EQ.1) GO TO 1
      A=1/(I*I+1)
      RETURN
1     A=(-1)/(I*I+1)
      RETURN
      END
```

Суммирование функциональных рядов

$$S(x)=\sum_{i=1}^{\infty}a_i(x)$$

в области сходимости x, с точки зрения программирования, приводит к незначительным усложнениям по сравнению с числовыми рядами. Действительно, пусть требуется найти $S(x)$ в точках x_j, $1\leqslant j\leqslant m$, из области сходимости с заданной абсолютной точностью ε. Тогда эта задача сводится к суммированию m числовых рядов

$$S(x_j)=\sum_{i=1}^{\infty}a_i(x_j)$$

с точностью ε, т. е. к задаче, рассмотренной выше. Поэтому приведенная программа является небольшой модификацией A5S0:

```
      SUBROUTINE A5S1(X,S,E,A,E1,M)
      REAL X(M),S(M),E
      INTEGER N,M
C     ВХОДНЫЕ ПАРАМЕТРЫ
C     X(M)—МАССИВ, СОДЕРЖАЩИЙ ЗНАЧЕНИЯ X,В
```

```
C            КОТОРЫХ ВЫПОЛНЯЕТСЯ СУММИРОВАНИЕ
C            РЯДА
C      E     —ЗАДАННАЯ АБСОЛЮТНАЯ ПОГРЕШНОСТЬ
C      A     —ИМЯ ВНЕШНЕЙ ФУНКЦИИ, ВЫЧИСЛЯЮЩЕЙ
C            ОБЩИЙ ЧЛЕН РЯДА, ДОЛЖНА ИМЕТЬ ВИД
C      REAL FUNCTION A(I,X)
C      REAL X
C      INTEGER I
C      E1    —ИМЯ ВНЕШНЕЙ ФУНКЦИИ, ВЫЧИСЛЯЮЩЕЙ
C            ДОСТИГНУТУЮ ПОГРЕШНОСТЬ, ДОЛЖНА
C            ИМЕТЬ ВИД
C      REAL FUNCTION E1(I,X)
C      REAL X
C      INTEGER I
C      M     —РАЗМЕРНОСТЬ МАССИВА X,S
C      ВЫХОДНЫЕ ПАРАМЕТРЫ
C      S(M)—МАССИВ, СОДЕРЖАЩИЙ СУММЫ РЯДА В
C            ТОЧКАХ X(J)
       DO 3 J=1,M
       N=1
       S(J)=0.
1      S(J)=S(J)+A(N,X(J))
       IF (E1(N,X(J)).GT.E) GO TO 2
       GO TO 3
2      N=N+1
       GO TO 1
3      CONTINUE
       RETURN
       END
```

Отличие этой программы от A5S0 состоит в том, что добавлен цикл по J — перебор по точкам x_j — и предполагается, что пользователь должен написать внешние функции-подпрограммы A(I,X), E1(I,X).

Применим программу A5S1 для суммирования ряда

$$S(x)=2\left[x+\frac{x^3}{3}+\frac{x^5}{5}+\ldots+\frac{x^{2n+1}}{2n+1}+\ldots\right],$$

с абсолютной погрешностью $\varepsilon=10^{-5}$ в точках $x_j=-0{,}5;\ -0{,}3;\ +0{,}4;\ +0{,}7$. Известно, что сумма этого ряда

$$S(x)=\ln\left((1+x)/(1-x)\right);$$

область сходимости $|x|<1$, оценка остатка

$$|S(x)-S_n(x)|\leqslant\frac{|x|^{2n+1}}{(2n+1)(1-x^2)}.$$

Программа может иметь вид

```
       REAL X(4),S(4),E
       INTEGER M
       EXTERNAL A,E1
```

```
      DATA E, M/1.E-5, 4/,X/-0.5,-0.3,0.4,0.7/
      CALL A5S1(X,S,E,A,E1,M)
      WRITE (5,1) S
1     FORMAT (2X,4E12.5)
      END
C     ...
      FUNCTION A(I,X)
      REAL X
      INTEGER I
      IF (MOD(I,2).EQ.1) GO TO 1
      A=0.
      RETURN
1     I=2*I+1
      A=2*X**I/I
      RETURN
      END
C     ...
      FUNCTION E1(I,X)
      REAL X
      INTEGER I
      IF (MOD(I,2).EQ.1) GO TO 1
      E1=1.E10
      RETURN
1     I=2*I+1
      E1=(ABS(X))**I/(I*(1-X*X))
      RETURN
      END
```

На примере этой программы можно заметить, что она далеко не универсальна. Действительно, если в массив x_j внести значение $x=1$, 3, лежащее вне области сходимости, то программа A5S1 зацикливается (нет выхода из цикла операторов с метками $1-3$) и заканчивается аварийно переполнением. Однако если пользователь знает область сходимости ряда и может оценить скорость сходимости, то применение программы A5S1 может быть оправдано.

С другой стороны, программа A5S1 служит для суммирования рядов общего вида, поэтому она более громоздка, чем программа суммирования конкретного ряда, например, приведенного выше.

3.3.6. Фурье-анализ. Под Фурье-анализом обычно понимается процедура определения коэффициентов ряда Фурье функции $f(x)$. В зависимости от того, в каком виде задается $f(x)$ (аналитически или таблично, в вещественном виде или комплексном) существует несколько процедур Фурье-анализа. Однако в основе всех процедур лежит вычисление интегралов по формулам

$$a_m=(1/\pi)\int_{-\pi}^{\pi} f(x)\cos mx\,dx, \quad m=0,\ 1,\ 2,\ \ldots,$$

$$b_m=(1/\pi)\int_{-\pi}^{\pi} f(x)\sin mx\,dx, \quad m=1,\ 2,\ \ldots.$$

Будем предполагать, что для вычисления $f(x)$ может быть написана функция-подпрограмма. Количество коэффициентов a_m, b_m, подлежащих определению, задается параметром M, $0 \leqslant m \leqslant M$. Для вычисления интеграла воспользуемся программой A4S0 с числом отрезков интегрирования $n = 2k(m+1)$, где коэффициент k будем задавать:

```
      SUBROUTINE A6S0(F,M,M1,K,A,B)
      REAL A(M1),B(M),PI,Q,R1,R2,R3
      INTEGER M,M1,K,I1,N
C     ------------------------------------------------------------
C     ВХОДНЫЕ ПАРАМЕТРЫ
C     F — ИМЯ ВНЕШНЕЙ ФУНКЦИИ-ПОДПРОГРАММЫ,
C         ВЫЧИСЛЯЮЩЕЙ F(X)
C     M — РАЗМЕРНОСТЬ МАССИВА B
C     M1 — РАЗМЕРНОСТЬ МАССИВА A, M1=M+1
C     K — КОЭФФИЦИЕНТ, ОПРЕДЕЛЯЮЩИЙ ЧИСЛО
C         ОТРЕЗКОВ ИНТЕГРИРОВАНИЯ
C     ВЫХОДНЫЕ ПАРАМЕТРЫ
C     A — МАССИВ КОЭФФИЦИЕНТОВ ФУРЬЕ A(I)
C     B — МАССИВ КОЭФФИЦИЕНТОВ ФУРЬЕ B(I)
C     ------------------------------------------------------------
      DATA PI/3.1415929/
C     ВЫЧИСЛЕНИЕ МАССИВА A
      H=PI/(K*M1)
      N=2*K*M1
      DO 2 I=1,M1
      Q=0.
      X=-PI
      I1=I-1
      DO 1 J=2,N,2
      R1=F(X)*COS(I1*X)
      R2=F(X+H)*COS(I1*(X+H))
      R3=F(X+2.*H)*COS(I1*(X+2.*H))
1     Q=Q+R1+4.*R2+R3
2     A(I)=Q*H/(3.*PI)
C     ВЫЧИСЛЕНИЕ МАССИВА B
      DO 4 I=1,M
      Q=0.
      X=-PI
      DO 3 J=2,N,2
      R1=F(X)*SIN(I*X)
      R2=F(X+H)*SIN(I*(X+H))
      R3=F(X+2.*H)*SIN(I*(X+2.*H))
3     Q=Q+R1+4.*R2+R3
4     B(I)=Q*H/(3.*PI)
      RETURN
      END
```

Так как в программе A6S0 отсутствует контроль точности вычисления интегралов, а следовательно, и значений a_m, b_m, то необходимо будет дважды обратиться к A6S0 с различными значениями k и $2k$ и воспользоваться для оценки погрешности правилом Рунге, например

```
      CALL A6S0(F,10,11,10,A,B)
      CALL A6S0(F,10,11,20,A,B)
```

Погрешность (массивы EA(I), EB(I)) затем определяется в циклах типа

```
      DO 1 I=1,M1
1     EA(I)=ABS(A1(I)-A(I))/15
```

Алгоритмы Фурье-анализа при больших значениях m (и соответственно n), основанные на общих квадратурных формулах, например формуле Симпсона, оказываются неэффективными.

В этом случае применяются разработанные специальные быстрые алгоритмы, учитывающие специфику тригонометрических функций. Эта группа алгоритмов обычно называется быстрым преобразованием Фурье. Если обычные алгоритмы имеют порядок роста числа арифметических операций $O(m^2)$, $m\to\infty$, то быстрые алгоритмы имеют порядок роста $O(m\log_2 m)$, $m\to\infty$, что указывает на их более высокую эффективность. Текст фортран-программы быстрого Фурье-преобразования приводится, например, в [30, с. 166].

3.3.7. Численное дифференцирование. Пусть функция $y(x)$ задана таблично с постоянным шагом h

$$y(x_i),\quad 1\leqslant i\leqslant n,\quad x_{i+1}-x_i=h.$$

Тогда для приближенного вычисления первой и второй производной можно использовать формулы

$$\frac{dy}{dx}(x_i)\simeq\frac{y(x_{i+1})-y(x_{i-1})}{2h},\quad 2\leqslant i\leqslant n-1,$$

$$\frac{d^2y}{dx^2}(x_i)\simeq\frac{y(x_{i+1})-2y(x_i)+y(x_{i-1})}{h^2},\quad 2\leqslant i\leqslant n-1.$$

Программа вычисления производных $y(x)$ по приведенным выше формулам является простой иллюстрацией использования операторов цикла:

```
      SUBROUTINE A7S0(N,N2,Y,H,D1,D2)
      REAL Y(N),D1(N2),D2(N2),H,H1,H2
      INTEGER N, N2
C     ---------------------------------------------------
C     ВХОДНЫЕ ПАРАМЕТРЫ
```

```
C       N — РАЗМЕРНОСТЬ МАССИВА Y
C       N2 — РАЗМЕРНОСТЬ МАССИВА D1,D2,N1=N-1
C       Y — МАССИВ, СОДЕРЖАЩИЙ ЗНАЧЕНИЯ ФУНКЦИИ
C       ВЫХОДНЫЕ ПАРАМЕТРЫ
C       D1 — МАССИВ, СОДЕРЖАЩИЙ ЗНАЧЕНИЯ ПЕРВОЙ
C            ПРОИЗВОДНОЙ
C       D2 — МАССИВ, СОДЕРЖАЩИЙ ЗНАЧЕНИЯ ВТОРОЙ
C            ПРОИЗВОДНОЙ
C       ------------------------------------------
        H1=2.*H
        H2=H*H
        DO 1 I=2,N1
        D1                  Y(I-1))/H1
1                             Y(I)+Y(I-1))/H2
        RETURN
        END
```

3.3.8. Операции с матрицами и векторами. В этом пункте приводятся некоторые программы операций с матрицами и векторами.

1) *Сложение матриц*: $c_{i,j}=a_{i,j}+b_{i,j}$, $1\leqslant i\leqslant m$, $1\leqslant j\leqslant n$.

```
        SUBROUTINE A8S0(A,B,M,N,C)
        REAL A(M,N),B(M,N),C(M,N)
        INTEGER M,N
        DO 1 I=1,M
        DO 1 J=1,N
1       C(I,J)=A(I,J)+B(I,J)
        RETURN
        END
```

2) *Умножение матриц*: $c_{ik}=\sum_{k=1}^{m} a_{i,j}b_{j,k}$, $1\leqslant i\leqslant l$, $1\leqslant k\leqslant n$.

```
        SUBROUTINE A8S1(A,B,L,M,N,C)
        REAL A(L,M),B(M,N),C(L,N),S
        INTEGER M,N,L
        DO 2 I=1,L
        DO 2 K=1,N
        S=0.
        DO 1 J=1,M
1       S=S+A(I,J)*B(J,K)
2       C(I,K)=S
        RETURN
        END
```

3) *Транспонирование квадратной матрицы* A *с размещением транспонированной* A' *на месте* A: $a'_{i,j}=a_{j,i}$, $1\leqslant i,\ j\leqslant n$.

```
        SUBROUTINE A8S2(A,N)
        REAL A(N,N),S
```

```
      INTEGER N,N1,I1
      N1=N-1
      DO 1 I=1,N1
      I1=I+1
      DO 1 J=I1,N
      S=A(I,J)
      A(I,J)=A(J,I)
1     A(J,I)=S
      RETURN
      END
```

4) *Транспонирование прямоугольной матрицы A переходом к одномерным массивам.*

```
      SUBROUTINE A8S3(A,M,N,MN,A1)
      REAL A(MN),A1(MN)
      INTEGER M,N,MN,L1,L2
C     MN — РАЗМЕРНОСТЬ ОДНОМЕРНЫХ МАССИВОВ,
C          СОДЕРЖАЩИХ МАТРИЦУ A
C          И ТРАНСПОНИРОВАННУЮ A1
      DO 1 I=1,M
      DO 1 J=1,N
      L1=(I-1)*N+J
      L2=(J-1)*M+I
1     A1(L2)=A(L1)
      RETURN
      END
```

5) *Скалярное произведение вектора x на вектор y*

$$S=\sum_{i=1}^{n} x_i y_i.$$

```
      REAL FUNCTION A8S4(N,X,Y)
      REAL X(N),Y(N),S
      INTEGER N
      S=0.
      DO 1 I=1,N
1     S=S+X(I)*Y(I)
      A8S4=S
      RETURN
      END
```

6) *Вычисление нормы матрицы A*

$$\| A \| = \max_{1 \leqslant i \leqslant n} \sum_{j=1}^{n} | a_{i,j} |.$$

```
      REAL FUNCTION A8S5(N,A,S)
      REAL A(N,N),S(N),R
C     S — РАБОЧИЙ МАССИВ
      INTEGER N
```

```
      DO 1 I=1, N
      R=0.
      DO 1 J=1, N
      R=R+ABS(A(I, J))
1     S(I)=R
C     ВЫБОР МАКСИМАЛЬНОГО ЭЛЕМЕНТА ИЗ S(I)
      R=S(1)
      DO 2 I=2, N
      IF (R.LT.S(I)) R=S(I)
2     CONTINUE
      A8S5=R
      RETURN
      END
```

3.3.9. Решение систем линейных алгебраических уравнений. Одним из методов решений системы уравнений вида

$$x = Bx + d,$$

где норма матрицы B меньше 1, является метод простой итерации (см. гл. 8)

$$x^{(k+1)} = Bx^{(k)} + d, \quad k = 0, 1, 2, \ldots, x^{(0)} = 0.$$

Итерации прекращаются, когда

$$\varepsilon_1 = \| x^{(k+1)} - x^{(k)} \| \leqslant \frac{(1 - \| B \|)}{\| B \|} \varepsilon,$$

где ε — заданная точность приближения к решению, норма вектора $\| x \| = \max\limits_{1 \leqslant i \leqslant n} |x_i|$. Программа может иметь следующий вид (для сокращения программы условие прекращения итераций принято в виде $\varepsilon_1 < \varepsilon$):

```
      SUBROUTINE A9S0(N,B,D,E,X,Y)
      REAL B(N,N),D(N),X(N),Y(N),E,S
      INTEGER N
C     ВХОДНЫЕ ПАРАМЕТРЫ
C     N — РАЗМЕРНОСТЬ МАССИВОВ B,D,X,Y
C     B — МАССИВ, СОДЕРЖАЩИЙ ЭЛЕМЕНТЫ МАТРИЦЫ
C         B
C     D — МАССИВ, СОДЕРЖАЩИЙ ЭЛЕМЕНТЫ СВОБОД-
C         НОГО ВЕКТОРА
C     E — ТОЧНОСТЬ
C     Y — РАБОЧИЙ МАССИВ
C     ВЫХОДНЫЕ ПАРАМЕТРЫ
C     X — ПРИБЛИЖЕННОЕ РЕШЕНИЕ
C     ---------------------------------------
C     ПОЛУЧЕНИЕ ПЕРВОГО ПРИБЛИЖЕНИЯ X
      DO 1 I=1, N
1     X(I)=D(I)
```

```
C       УМНОЖЕНИЕ МАТРИЦЫ В НА ВЕКТОР X
100     DO 3 I=1,N
        S=0.
        DO 2 J=1, N
2       S=S+B(I,J)*X(J)
3       Y(I)=S
C       СЛОЖЕНИЕ BX+D—ПОЛУЧЕНИЕ НОВОГО ПРИ-
C                     БЛИЖЕНИЯ
        DO 4 I=1, N
4       Y(I)=Y(I)+D(I)
C       ОПРЕДЕЛЕНИЕ НОРМЫ РАЗНОСТИ Y—X
        S=ABS(Y(1)-X(1))
        DO 5 I=2, N
        IF (S.LT.ABS(Y(I)-X(I))) S=ABS(Y(I)-X(I))
5       CONTINUE
C       ЗАСЫЛКА НОВОГО ПРИБЛИЖЕНИЯ В СТАРОЕ
        DO 6 I=1, N
6       X(I)=Y(I)
C       СРАВНЕНИЕ С ЗАДАННОЙ ТОЧНОСТЬЮ
        IF (S.LT.E) GO TO 7
        GO TO 100
7       RETURN
        END
```

3.3.10. Решение линейных интегральных уравнений. Одним из численных методов решения интегрального уравнения

$$y(x)=\int_a^b K(x, s)\,y(s)\,ds+f(x), \qquad a\leqslant x\leqslant b,$$

является его приближенная замена системой линейных алгебраических уравнений

$$y(x_i)=\sum_{j=1}^{n} hK(x_i, s_j)\,y(s_j)+f(x_i), \qquad 1\leqslant i\leqslant n, \quad 1\leqslant j\leqslant n,$$

$$x_i=a+(i-1/2)h, \quad s_j=a+(j-1/2)h, \quad h=(b-a)/n.$$

Обозначим матрицу $K=(K_{i,j})=hK(x_i, s_j)$, векторы y, f

$$y=(y_i), \quad y_i=y(x_i), \quad f=(f_i), \quad f_i=f(x_i).$$

Будем предполагать, что $\|K\|\leqslant q<1$. Тогда систему алгебраических уравнений

$$y=Ky+f$$

можно решать с помощью программы A9S0, а центральным моментом следующей программы оказывается вычисление матрицы K и свободного вектора f. Значение элементов вектора y принимается в качестве приближенных значений точного решения $y(x)$ в точках x_i.

```
      SUBROUTINE B2S0(N,K,F,A0,B0,B,D,E,X,Y)
      REAL A0,B0,B(N,N),D(N),Y(N),X0,S,E,X(N)
      INTEGER N
C     ------------------------------------------------------------
C     ВХОДНЫЕ ПАРАМЕТРЫ
C      N — РАЗМЕРНОСТЬ МАССИВОВ B,D,Y
C      K — ВНЕШНЯЯ ФУНКЦИЯ-ПОДПРОГРАММА,
C          ВЫЧИСЛЯЮЩАЯ K(X,S)
C     REAL FUNCTION K(X,S)
C     REAL X,S
C     A0 — НИЖНИЙ ПРЕДЕЛ ИНТЕГРАЛА
C     B0 — ВЕРХНИЙ ПРЕДЕЛ ИНТЕГРАЛА
C     B  — РАБОЧИЙ МАССИВ
C     E  — ТОЧНОСТЬ РЕШЕНИЯ СИСТЕМЫ УРАВНЕНИЙ
C     X  — РАБОЧИЙ МАССИВ
C     ВЫХОДНЫЕ ПАРАМЕТРЫ
C     Y  — ВЕКТОР ПРИБЛИЖЕННЫХ ЗНАЧЕНИЙ
C          РЕШЕНИЯ
C     ------------------------------------------------------------
C     ВЫЧИСЛЕНИЕ ЭЛЕМЕНТОВ МАТРИЦЫ K(I,J),F(I) И
C     ЗАПИСЬ В МАССИВЫ B(N,N),D(N)
      H=(B0-A0)/N
      X0=A0
      S=A0
      DO 2 J=1,N
      DO 1 I=1,N
      B(I,J)=K(X0,S)
1     X0=X0+H/2
      D(J)=F(S)
2     S=S+H/2
C     ОБРАЩЕНИЕ К ПРОГРАММЕ A9S0
      CALL A9S0(N,B,D,E,Y,X)
      RETURN
      END
```

3.3.11. Корни полиномов и нелинейных уравнений. Существует множество алгоритмов определения корней нелинейных уравнений, в частности полиномов. Рассмотрим наиболее простой алгоритм, применимый для полиномов и скалярных уравнений вида

$$f(x)=0 \quad \text{или} \quad P_n(x)=0,$$

где $f(x)$ — непрерывная функций; $P_n(x)$ — полином n-й степени; x — скаляр, принадлежащий заданному интервалу $a \leqslant x \leqslant b$.

Первый этап: вычисление перебором значений $f(x)$ с шагом h_1 в точках $x_i=a+ih_1$, $1 \leqslant i \leqslant n$, $h_1=(b-a)/n$, вывод на терминал граничных точек интервалов $[x_i, x_{i+1}]$, где $f(x_i)f(x_{i+1})<0$ или $f(x_i)=0$, $f(x_{i+1})=0$. Указанные интервалы заведомо содержат по крайней мере один корень.

Второй этап: повторение первого этапа с более мелкими шагами на выделенных интервалах с целью обнаружения нескольких корней на интервалах $[x_i, x_{i+1}]$. Этот этап можно опустить, если пользователь имеет дополнительную информацию о расположении корней уравнений.

Третий этап: в предположении, что на заданном интервале уравнение имеет единственный корень, определение его с заданной точностью методом бисекции (см. **9.4**).

Первый и второй этапы реализуются в следующей программе:

```
      SUBROUTINE B4S0(A,B,H,F)
      REAL A,B,H,X1,X2,Y1,Y2,Y3,Y4
      INTEGER K1,K2,K3
      DATA K1,K2,K3,Y4/3*0, 1./
C     ------------------------------------------------------------
C     ВХОДНЫЕ ПАРАМЕТРЫ
C     A — ЛЕВЫЙ КОНЕЦ ИНТЕРВАЛА
C     B — ПРАВЫЙ КОНЕЦ ИНТЕРВАЛА
C     H — ШАГ ПЕРЕБОРА
C     F — ИМЯ ВНЕШНЕЙ ФУНКЦИИ-ПОДПРОГРАММЫ,
C         ВЫЧИСЛЯЮЩЕЙ ЗНАЧЕНИЯ F(X) ИЛИ P(X)
C     ------------------------------------------------------------
      X1=A
1     Y1=F(X1)
      X2=X1+H
      Y2=F(X2)
      Y3=SIGN(Y4, Y1)*SIGN(Y4, Y2)
      IF (Y1.EQ.0.) K1=1
      IF (Y2.EQ.0.) K2=1
      IF (Y3.LT.0.) K3=1
      IF ((K1.OR.K2.OR.K3).NE.0) GO TO 3
2     IF (X2.GT.B) GO TO 4
      X1=X1+H
      GO TO 1
3     WRITE (5, 5) X1, X2, K1, K2, K3
      GO TO 2
4     WRITE (5, 6)
5     FORMAT (2X, 'X1=', E13.6, 'X2=', E13.6, 2X, 3I2)
6     FORMAT (10X, 'КОНЕЦ ПЕРЕБОРА')
      RETURN
      END
```

Приведем программу метода бисекции по схеме алгоритма п. 9.4.2:

```
      SUBROUTINE B4S1(A,B,E,F,C)
      REAL A,B,E,C,S,Y,Y1
      DATA S/1./
C     ------------------------------------------------------------
```

```
C       ВХОДНЫЕ ПАРАМЕТРЫ
C       A — ЛЕВЫЙ КОНЕЦ ИНТЕРВАЛА
C       B — ПРАВЫЙ КОНЕЦ ИНТЕРВАЛА
C       E — ТОЧНОСТЬ ОПРЕДЕЛЕНИЯ КОРНЯ
C       F — ИМЯ ВНЕШНЕЙ ФУНКЦИИ-ПОДПРОГРАММЫ,
C           ВЫЧИСЛЯЮЩЕЙ ЗНАЧЕНИЯ F(X) ИЛИ P(X)
C       ВЫХОДНЫЕ ПАРАМЕТРЫ
C       C — ЗНАЧЕНИЕ КОРНЯ
C       ------------------------------------------------------------
1       C=(A+B)*0.5
        Y=F(C)
        IF (Y.EQ.0) GO TO 2
        Y1=F(A)
        IF (SIGN(S, Y)*SIGN(S, Y1).GT.0) A=C
        Y1=F(B)
        IF (SIGN(S, Y)*SIGN(S, Y1).GT.0) B=C
        IF ((B-A)*0.5.GE.E) GO TO 1
2       RETURN
        END
```

3.3.12. Нелинейная оптимизация. В основе многих алгоритмов поиска минимума (максимума) нелинейной целевой функции $\Phi(x)$ (x, вообще говоря, вектор) лежит минимизация функции одной переменной. Одним из возможных алгоритмов минимизации функции одной переменной является алгоритм метода золотого сечения (см. **9.7**).

Приведем программу, соответствующую схеме алгоритма метода золотого сечения (см. п. 9.7.3):

```
        SUBROUTINE B5S0(A,B,E,F,C)
        REAL A,B,E,C,T
C       ------------------------------------------------------------
C       ВХОДНЫЕ ПАРАМЕТРЫ
C       A — ЛЕВЫЙ КОНЕЦ ИНТЕРВАЛА
C       B — ПРАВЫЙ КОНЕЦ ИНТЕРВАЛА
C       E — ПОГРЕШНОСТЬ ОПРЕДЕЛЕНИЯ ТОЧКИ MIN
C       F — ИМЯ ВНЕШНЕЙ ФУНКЦИИ-ПОДПРОГРАММЫ,
C           ВЫЧИСЛЯЮЩЕЙ ЗНАЧЕНИЕ МИНИМИЗИРУЕ-
C           МОЙ ФУНКЦИИ
C       ВЫХОДНЫЕ ПАРАМЕТРЫ
C       C — ПРИБЛИЖЕННОЕ ЗНАЧЕНИЕ ТОЧКИ MIN
C       ------------------------------------------------------------
        DATA T/0.6180339/
        X1=B-(B-A)*T
        X2=A+(B-A)*T
        Y1=F(X1)
        Y2=F(X2)
1       IF (Y2.LT.Y1) GO TO 2
        B=X2
```

```
      X2=X1
      Y2=Y1
      X1=B-(B-A)*T
      Y1=F(X1)
      GO TO 3
2     A=X1
      X1=X2
```

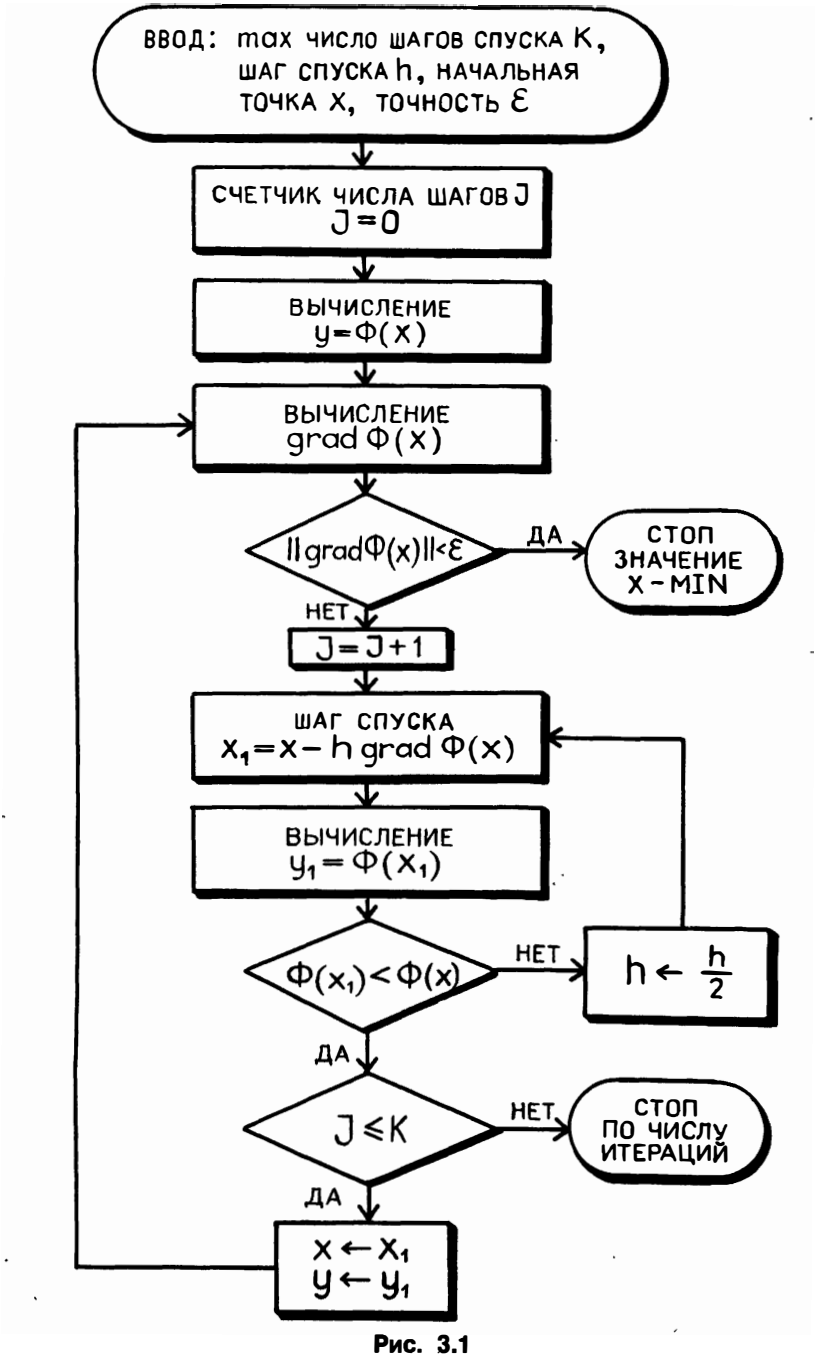

Рис. 3.1

```
      Y1=Y2
      X2=A+(B-A)*T
      Y2=F(X2)
3     IF ((B-A).GT.E) GO TO 1
      C=(A+B)*0.5
      RETURN
      END
```

Алгоритм метода градиентного спуска поиска минимума функции $\Phi(x_1, \ldots, x_n)$ нескольких переменных задается формулой (9.6.2). Дополним его процедурой дробления шага $(h/2)$, если $\Phi(x^{(k+1)}) \geqslant \Phi(x^k)$. Схема алгоритма изображена на рис. 3.1.

Программа, соответствующая схеме, приведенной на рис. 3.1, имеет следующий вид:

```
      SUBROUTINE B5S1(N,X,Y,G,H,F,F1,X1,J,K)
      REAL X(N),G(N),Y,H,E,X1(N),Y1,S
      INTEGER N,K,J
C     ВХОДНЫЕ ПАРАМЕТРЫ
C     N—РАЗМЕРНОСТЬ МАССИВОВ X,G
C     X—МАССИВ, СОДЕРЖИТ ЭЛЕМЕНТЫ НАЧАЛЬНОГО
C        ВЕКТОРА
C     H—НАЧАЛЬНЫЙ ШАГ СПУСКА
C     E—ТОЧНОСТЬ, ОПРЕДЕЛЯЮЩАЯ КОНЕЦ ПРОЦЕДУ-
C        РЫ СПУСКА
C     F—ИМЯ ВНЕШНЕЙ ПОДПРОГРАММЫ, ВЫЧИСЛЯ-
C        ЮЩЕЙ ЗНАЧЕНИЕ ЦЕЛЕВОЙ ФУНКЦИИ
C     SUBROUTINE F(X,Y)
C     REAL X(N),Y
C     F1—ИМЯ ВНЕШНЕЙ ПОДПРОГРАММЫ, ВЫЧИС-
C        ЛЯЮЩЕЙ ЗНАЧЕНИЕ ВЕКТОРА ГРАДИЕНТА ЦЕ-
C        ЛЕВОЙ ФУНКЦИИ
C     SUBROUTINE F1(X,G)
C     REAL X(N),G(N)
C     X1—РАБОЧИЙ МАССИВ
C     K—МАКСИМАЛЬНОЕ ЧИСЛО ШАГОВ СПУСКА
C     ВЫХОДНЫЕ ПАРАМЕТРЫ
C     Y—ЗНАЧЕНИЕ ЦЕЛЕВОЙ ФУНКЦИИ
C     G—ЗНАЧЕНИЕ ГРАДИЕНТА ЦЕЛЕВОЙ ФУНКЦИИ
C     X—МАССИВ, СОДЕРЖИТ ЭЛЕМЕНТЫ ВЕКТОРА,
C        МИНИМИЗИРУЮЩЕГО ЦЕЛЕВУЮ ФУНКЦИЮ
C     J—ЧИСЛО ШАГОВ СПУСКА
C     ..................................................
      J=0
      CALL F(X,Y)
      CALL F1(X,G)
C     ВЫЧИСЛЕНИЕ КВАДРАТА ЕВКЛИДОВОЙ НОРМЫ G
      S=0
      DO 2 I=1,N
```

```
2       S=S+G(I)*G(I)
C       ------------------------------------------------------------------
        IF (S.LT.E) GO TO 7
        J=J+1
C       ШАГ СПУСКА
3       DO 4 I=1,N
4       X1(I)=X(I)-H*G(I)
C       ------------------------------------------------------------------
        CALL F(X1,Y1)
        IF (Y1.LE.Y) GO TO 5
        H=H*0.5
        GO TO 3
5       IF (J.GT.K) GO TO 7
        DO 6 I=1,N
6       X(I)=X1(I)
        Y=Y1
        GO TO 1
7       RETURN
        END
```

В качестве иллюстрации применения программы B5S1 рассмотрим минимизацию целевой функции

$$\Phi(x_1,\ x_2)=(x_1-2)^4+x_2^2.$$

Положим максимальное число шагов спуска $k=100$, точность $\varepsilon=10^{-5}$, начальный вектор $x=(1,\ 2)$, начальный шаг $h=0{,}5$. Программа этой задачи может быть представлена в форме

```
        REAL X(2),Y,G(2),H,E,X1(2)
        INTEGER N,K,J
        EXTERNAL F,F1
        DATA X/1.,2./,E,H/1.E—5,0.5/,K,N/100,2/
        CALL B5S1(N,X,Y,G,H,E,F,F1,X1,J,K)
        WRITE (5,1) X,Y,J
1       FORMAT (2X,2E13.6,2X,E13.6,2X,I3)
        END
        SUBROUTINE F(X,Y)
        REAL X(2),Y
        Y=(X(1)—2.)**4+X(2)*X(2)
        RETURN
        END
        SUBROUTINE F1(X,G)
        REAL X(2),G(2)
        G(1)=4.*(X(1)—2.)**3
        G(2)=2.*X(2)
        RETURN
        END
```

3.3.13. Решение обыкновенных дифференциальных уравнений; задача Коши. Рассмотрим два алгоритма решения задачи Коши: метод Эйлера и метод Рунге—Кутта 4-го порядка. Один шаг метода Эйлера задается формулой (10.2.5). Пусть $y(x)$ — вектор с n компонентами, заданный в точке x; тогда один шаг интегрирования методом Эйлера — это вычисление вектора $y(x+h)$ в точке $x + h$ по формуле

$$y(x+h)=y(x)+hf(x,\ y(x)),$$

где $f(x,\ y)$ — вектор-функция правых частей дифференциального уравнения (в векторной форме)

$$\frac{dy}{dx}=f(x,\ y).$$

Программа одного шага метода Эйлера:

```
      SUBROUTINE B6S0(N,Y,X,H,R,D)
      REAL Y(N),X,H,D(N)
      INTEGER N
C     ВХОДНЫЕ ПАРАМЕТРЫ
C     N—РАЗМЕРНОСТЬ ВЕКТОРА Y,D
C     Y—МАССИВ, СОДЕРЖАЩИЙ ВЕКТОР Y(X)
C     X—НАЧАЛЬНОЕ ЗНАЧЕНИЕ АРГУМЕНТА
C     H—ШАГ ИНТЕГРИРОВАНИЯ
C     R—ИМЯ ВНЕШНЕЙ ПОДПРОГРАММЫ, ВЫЧИСЛЯ-
C        ЮЩЕЙ ПРАВЫЕ ЧАСТИ УРАВНЕНИЙ
C     SUBROUTINE R(X,Y,D)
C     REAL X,Y(N),D(N)
C     D—РАБОЧИЙ МАССИВ, СОДЕРЖИТ ПРАВЫЕ ЧАСТИ
C        УРАВНЕНИЙ
C     ВЫХОДНЫЕ ПАРАМЕТРЫ
C     X—ЗНАЧЕНИЕ АРГУМЕНТА X+H
C     Y—МАССИВ, СОДЕРЖАЩИЙ ВЕКТОР Y(X+H)
C     .....
      CALL R(X,Y,D)
      DO 1 I=1,N
1     Y(I)=Y(I)+H*D(I)
      X=X+H
      RETURN
      END
```

Применим программу B6S0 для интегрирования методом Эйлера с постоянным шагом $h=0{,}1$, на интервале $0\leqslant x\leqslant 5$ системы дифференциальных уравнений

$$\frac{dy_1}{dx}=\sin(y_1+y_2),\ \ y_1(0)=1,$$

$$\frac{dy_2}{dx}=\cos(y_1+xy_2),\ \ y_2(0)=2.$$

Чтобы решить эту задачу, следует выполнить 50 шагов (50 обращений к B6S0) для получения векторов

$$\begin{pmatrix} y_1(0,\,1) \\ y_2(0,\,1) \end{pmatrix},\quad \begin{pmatrix} y_1(0,\,2) \\ y_2(0,\,2) \end{pmatrix},\quad \ldots,\quad \begin{pmatrix} y_1(5,\,0) \\ y_2(5,\,0) \end{pmatrix};$$

«стартуя» с начального вектора, на каждом 5-м шаге будем выводить на терминал значения $(x,\ y_i(x))$. Соответствующая программа:

```
      REAL Y(2),X,H,D(2)
      INTEGER K,N,M
      EXTERNAL R
      DATA X/0./,Y/1.,2./,H/0.1/,K,N/50,2/
      DO 1 I=1,K
      CALL B6S0(N,Y,X,H,R,D)
      M=MOD(I,5)
      IF (M.NE.0) GO TO 1
      WRITE (5,2) X,Y
1     CONTINUE
2     FORMAT (2X,'X=',E9.2,2X,2E13.6)
      END
      SUBROUTINE R(X,Y,D)
      REAL X,Y(2),D(2)
      D(1)=SIN(Y(1)+Y(2))
      D(2)=COS(Y(1)-X*Y(2))
      RETURN
      END
```

По аналогии с программой B6S0 составим программу одного шага метода Рунге—Кутта 4-го порядка, который задается для скалярного случая формулой (10.2.12). Описание входных и выходных параметров здесь почти такое же, как в B6S0, поэтому текст программы приводится без дополнительных комментариев:

```
      SUBROUTINE B6S1(N,Y,X,H,R,D)
      REAL Y(N),X,H,D(N),K(4,N),Y1(N)
      INTEGER N
C     K(4,N)—РАБОЧИЙ МАССИВ, СОДЕРЖИТ
C           КОЭФФИЦИЕНТЫ К1, К2, К3, К4
C     Y1(N)—РАБОЧИЙ МАССИВ
C     ..........................................................................
      CALL R(X,Y,D)
      DO 1 I=1,N
      K(1,I)=H*D(I)
      Y1(I)=Y(I)+0.5*K(1,I)
      X=X+0.5*H
      CALL R(X,Y1,D)
      DO 2 I=1,N
      K(2,I)=H*D(I)
2     Y1(I)=Y(I)+0.5*K(2,I)
```

```
      CALL R(X,Y1,D)
      DO 3 I=1,N
      K(3,I)=H*D(I)
3     Y1(I)=Y(I)+K(3,I)
      X=X+0.5*H
      CALL R(X,Y1,D)
      DO 4 I=1,N
      K(4,I)=H*D(I)
4     Y(I)=Y(I)+0.166667*(K(1,I)+2.*K(2,I)+2.*K(3,I)+K(4,I))
      RETURN
      END
```

Чтобы проиллюстрировать применение программы B6S1 для интегрирования с постоянным шагом $h=0{,}1$ на интервале $0 \leqslant x \leqslant 5$ приведенной выше системы уравнений, следует сделать всего лишь одно исправление — заменить B6S0 на B6S1.

3.3.14. Краевая задача; метод прогонки. В основе решения линейных краевых задач для уравнений второго порядка разностными методами лежит алгоритм решения трехдиагональных систем линейных алгебраических уравнений — метод прогонки (см. п. 10.4.2). Будем предполагать, что прогонка для заданной трехдиагональной системы осуществима и устойчива (см. теорему 10.6). В общем виде трехдиагональная система записывается в форме

$$A_i y_{i-1} + C_i y_i + B_i y_{i+1} = R_i; \quad 1 \leqslant i \leqslant n, \quad A_1 = B_n = 0,$$

где коэффициенты A_i, B_i, C_i, R_i (заданные массивы) — входные параметры, y_i (искомый вектор) — выходной параметр. Формулы прогонки можно представить в следующей форме:

прямая прогонка:

$$\alpha_1 = 0, \quad \beta_1 = 0,$$

$$\alpha_i = \frac{B_{i-1}}{-(C_{i-1} + A_{i-1}\alpha_{i-1})}, \quad \beta_i = \frac{A_{i-1}\beta_{i-1} - R_{i-1}}{-(C_{i-1} + A_{i-1}\alpha_{i-1})},$$

обратная прогонка:

$$y_n = \frac{A_n\beta_n - R_n}{-(C_n + A_n\alpha_n)}, \quad y_{j-1} = \alpha_j y_j + \beta_j, \quad n \leqslant j \leqslant 2.$$

В следующей программе реализованы формулы прогонки, коэффициенты α_i, β_i размещаются в массивах E, D соответственно:

```
      SUBROUTINE A9S1(N,A,B,C,R,E,D,Y)
      REAL A(N),B(N),C(N),R(N),E(N),D(N),Y(N),S
      INTEGER N, J
C     ПРЯМАЯ ПРОГОНКА
      E(1)=0.
      D(1)=0.
      DO 1 I=2,N
      S=-(C(I-1)+A(I-1)*E(I-1))
```

```
      E(I)=B(I-1)/S
1     D(I)=(A(I-1)*D(I-1)-R(I-1))/S
C     ОБРАТНАЯ ПРОГОНКА
      Y(N)=(A(N)*D(N)-R(N))/(-C(N)+A(N)*E(N))
      DO 2 I=2,N
      J=N+2-I
2     Y(J-1)=E(J)*Y(J)+D(J)
      RETURN
      END
```

Рассмотрим разностную схему краевой задачи (10.4.9) с более простыми, чем (10.4.10), условиями

$$z_0=0, \quad z_m=0,$$

$$(z_{i+1}-2z_i+z_{i-1})/h^2+q_iz_i=f_i, \quad 1\leqslant i\leqslant m+1.$$

Здесь $q_i\leqslant 0$, $q_i=q(x_i)$, $f_i=f(x_i)$, $x_i=a+ih$, $h=(b-a)/m$.

Программа решения этой разностной схемы состоит из вычисления элементов массива A, B, C, R и обращения к программе A9S1. Имеем соответствие

$$y_i=z_{i-1}, \quad 1\leqslant i\leqslant m+1; \quad n=m+1, \quad n\geqslant 2;$$

$$A_i=1/h^2, \ B_i=1/h^2, \ C_i=-2/h^2+q_{i-1}, \ R_i=f_{i-1}, \ 2\leqslant i\leqslant n-1.$$

Краевые условия $z_0=z_m=0$ приводят к равенствам

$$A_1=B_1=R_1=0, \ C_1=1; \ A_n=B_n=R_n=0, \ C_n=1.$$

Входными параметрами задачи являются: n — размерность вектора y, левый a и правый b концы интервала решения краевой задачи, формула функций $q(x)$, $f(x)$ — коэффициента и правой части дифференциального уравнения.

```
      SUBROUTINE B7S0(N,A0,B0,Q,F,A,B,C,R,E,D,Y)
     *REAL A0,B0,A(N),B(N),C(N),R(N),E(N),D(N),Y(N),
      H,H2,X
      INTEGER N
C     A0,B0 — ЛЕВЫЙ, ПРАВЫЙ КОНЦЫ ИНТЕРВАЛА
C             РЕШЕНИЯ
C     Q,F   — ИМЕНА ВНЕШНИХ ФУНКЦИЙ-ПОДПРО-
C             ГРАММ, ВЫЧИСЛЯЮЩИХ Q(X), F(X) СООТ-
C             ВЕТСТВЕННО
C     ВЫЧИСЛЕНИЕ ВХОДНЫХ ПАРАМЕТРОВ ПРО-
C     ГРАММЫ A9S1
      A(1)=0.
      B(1)=0.
      R(1)=0.
      C(1)=1.
      A(N)=0.
      B(N)=0.
      R(N)=0.
      C(N)=1.
```

```
      H=(B0-A0)/(N-1)
      H2=H*H
      DO 1 I=2,N-1
      X=A0+(I-1)*H
      A(I)=1./H2
      B(I)=1./H2
      C(I)=Q(X)-2./H2
1     R(I)=F(X)
C     ОБРАЩЕНИЕ К ПРОГРАММЕ A9S1
      CALL A9S1(N,A,B,C,R,E,D,Y)
      RETURN
      END
```

● 3.4. Оптимизация программ

3.4.1. Введение. Программирование занимает в технологии вычислительного процесса определенное место. Оптимизация всего цикла вычислений от составления математической модели до получения числового результата — очень сложная задача. Под оптимизацией понимается, например, минимизация или максимизация некоторого критерия — целевой функции Ф. Критерием может быть стоимость, надежность, быстродействие вычислений. Решение задачи оптимизации необходимо проводить сразу по всей совокупности возможных элементов вычислительного процесса, т. е. искать оптимальные математическую модель (М), вычислительный алгоритм (ВАЛ), программу (П) для заданной вычислительной системы. Если есть возможность выбирать ЭВМ, то в число свободных элементов, подлежащих определению, включается архитектура ЭВМ (АР). Если имеется выбор операционной системы, то появляется еще свободный параметр — тип операционной системы (ОС).

Таким образом, для решаемой задачи может быть, в принципе, определена зависимость критерия качества решения задачи Ф от применяемых элементов вычислительного процесса

$$\Phi = \Phi(M,\ ВАЛ,\ П,\ ОС,\ АР)$$

и тогда задача оптимизации состоит в определении $\min\limits_{D}\Phi$ или $\max\limits_{D}\Phi$ на множестве D возможных значений элементов M, $ВАЛ$, $П$, $ОС$, $АР$. Решение этой, как уже отмечалось выше, очень сложной задачи выходит за рамки данной книги. Однако следует всегда иметь в виду, что улучшение только программы или только вычислительного алгоритма представляет собой фактически попытку оптимизации Ф по одному элементу при фиксированных остальных. Поэтому и результат может быть весьма далек от оптимального по полному набору элементов.

В нашем подходе к улучшению программ мы будем исходить из того, что из определенного опыта в предметной области уже выбрана математическая модель, а из анализа методов вычислений выбран вычислительный алгоритм. Выбранный алгоритм либо уже запрограммирован, либо может быть запрограммирован на фортране. Так как программа на фортране не может быть выполнена на ЭВМ, фортран-программа преобразуется (транслируется) на машинный язык. Программа на машинном языке называется объектной программой или объектным кодом.

Основными критериями эффективности программы будем считать время выполнения и объем оперативной памяти для реализации объектного кода. При этом часто время выполнения можно сократить увеличивая объем необходимой памяти. Поскольку для решения средних по сложности задач объем оперативной памяти не представляет ограничений, мы в целях улучшения программ в основном будем стремиться к их максимальному быстродействию.

Наконец, необходимо отметить, что существующие трансляторы с фортрана имеют оптимизирующие блоки. Однако не следует думать, что неэффективную программу такой транслятор преобразует в эффективную. Он лишь улучшит некоторые доступные ему фрагменты программы. Поэтому для написания эффективных программ пользователь должен самостоятельно по ходу программирования осуществлять оптимизирующие преобразования. Более подробно с оптимизацией программ можно познакомиться в [17].

3.4.2. Исключение повторных вычислений. Это преобразование предусматривает исключение из программы избыточных команд. С этой целью, просматривая текст фортран-программы, исключают выражения, которые производят выполненные ранее вычисления.

Например, в последовательности операторов

```
X=A*(1.+COS(B))
Y=A-C/B
Z=C/B+1.+COS(B)
```

слагаемые

```
C/B+(1.+COS(B))
```

в последнем операторе производят уже выполненные ранее вычисления. Поэтому вместо приведенного фрагмента следует использовать более эффективную последовательность операторов

```
U=1.+COS(B)
V=C/B
X=A*U
Y=A-V
Z=V+U
```

Хотя новый фрагмент длиннее старого, но он эффективнее по времени выполнения и объему используемой памяти.

Следующий, более сложный, пример указывает на то, что рассматриваемое преобразование необходимо выполнять осторожно, так как присутствие в программе похожих выражений не означает, что они вычисляют одинаковые значения. Например, в последовательности

```
X=Y*Z-U
V=Y
Y=Y+2.-Y*Z
W=Y*Z+Z*V
```

нельзя заменить Y*Z одной переменной, поскольку в третьем операторе происходит изменение значения Y. С другой стороны, в первом и четвертом операторах непохожее выражение Z*V на Y*Z вычисляет (из-за второго оператора) то же самое значение. Здесь повторные вычисления исключаются следующим образом:

```
T=Y*Z
X=T-U
V=Y
Y=Y+2.-T
W=Y*Z+T
```

3.4.3. Замена медленно выполняемых операций на более быстрые. Время выполнения арифметических операций различно, оно возрастает в следующем порядке:

сложение или вычитание,
умножение,
деление,
возведение в степень.

Поэтому для небольших целых чисел N выражения вида

```
A*N; A**N
```

целесообразно заменять на

```
A+A+...+A; A*A*...*A
```

одновременно исключая повторные вычисления. Операцию деления целесообразно заменять умножением на обратную величину. Поясним эти рекомендации на примере:

```
C=A*2+B;                    C=A+A+B
D=X**2+(X**3+4.)/Y;         D=X*X+(X*X*X+4.)/Y
E=(A+B)*(1.+X**4);          E=(A+B)*(1.+X*X*X*X)
```

где слева — фрагмент программы до преобразования, справа — после него. Затем из правого фрагмента, исключая повторные вычисления, получим более эффективную программу

```
T=A+B
C=A+T
S=X*X
R=S*X
```

```
D=S+(R+4)/Y
E=T*(1+R*X)
```

Замена операции деления иллюстрируется следующим примером:

```
X=A/Y;                    T=1./Y
Z=(A+B)/Y+SIN(1./Y);      X=A*T
V=Z/Y;                    Z=(A+B)*T+SIN(T)
```

или

```
                          V=Z*T
A=B/2;                    A=B*0.5
```

3.4.4. Применение констант в арифметических выражениях. Если значения некоторых переменных в процессе вычислений не изменяются и они известны, то замена переменных константами в арифметических выражениях позволит увеличить быстродействие. Однако, как уже упоминалось выше, изменение значения такой константы может потребовать значительных усилий при исправлении текста программы.

Пример.

```
A=1;                      A=1
B=2;                      B=2
C=(A+B)*2+Z;              C=6+Z
D=A*A-Z;                  D=1-Z
```

Если необходимо будет заменить значение A=1 другой константой, то в старом варианте следует исправить один оператор, а в новом — три.

3.4.5. Сокращение преобразований данных. При вычислении арифметических выражений, в которых участвуют переменные и константы разных типов (целые, вещественные, с двойной точностью, комплексные), происходит преобразование данных к одному типу, а затем выполнение операций. Поэтому, если позаботиться о том, чтобы в вычислениях как можно меньше смешивались типы данных, можно ускорить выполнение программы. Например, замена оператора

```
A=1
```

следующим

```
A=1.0
```

исключает преобразование данных, если A имеет вещественный тип.

3.4.6. Сокращение индексированных переменных. Любое обращение к индексированной переменной (элементам массива) связано с выполнением команд, вычисляющих значение индекса и адрес переменной. Время на выполнение операции можно сократить, если удастся заменить в некоторых операторах индексированные переменные на переменные без индекса. Например, замена левого цикла на правый

```
      DO 1 I=1,100;                  DO 1 I=1,100
      A(I)=X(I)+COS(X(I));           T=X(I)
      B(I)=X(I)+SIN(X(I));           A(I)=T+COS(T)
1     CONTINUE;                    1 B(I)=T+SIN(T)
```

в четыре раза сокращает время на вычисление адреса переменной X(I).

Однако этот пример может привести и к отрицательному эффекту: если вычисления производятся на ЭВМ с векторным процессором (так проявляется оптимизация без учета архитектуры ЭВМ), то левый цикл, как правило, будет предпочтительнее правого.

В дальнейшем рекомендации по преобразованию программ относятся к ЭВМ традиционной архитектуры с последовательным выполнением команд.

3.4.7. Улучшение структуры циклов. 1) Объединение циклов сокращает время на управление операторами цикла и необходимый объем памяти. Приведем два примера объединения.

Первый пример объединенные циклы

```
      DO 1 I=1,100;                  DO 1 I=1,100
1     A(I)=1.;                       A(I)=1.
      DO 2 J=1,100;                  B(I)=X(I)
2     B(J)=X(J);                   1 CONTINUE
```

Второй пример

```
      DO 1 I=1,100;                  DO 1 I=1,100
1     A(I)=1.;                       A(I)=1.
      DO 2 J=1,150;                1 B(I)=X(I)
      B(J)=X(J);                     DO 2 J=101,150
2     CONTINUE;                    2 B(J)=X(J)
```

2) Развертка цикла сокращает время на организацию цикла. Например, развертка левого цикла в правый сокращает вдвое число операций приращения управляющей переменной и контроля на завершение:

```
      DO 1 I=1,100;                  DO 1 I=1,99,2
      A(I)=1.;                       A(I)=1.
1     CONTINUE;                    1 A(I+1)=1.
```

Но при этом увеличивается необходимый объем памяти.

3) Вывод из цикла неизменяемых выражений производится, если эти выражения постоянны внутри цикла. Например, замена

```
      DO 1 I=1,1000;                 S=R*T
      A(I)=X(I)+R*T;                 DO 1 I=1,1000
1     CONTINUE;                    1 A(I)=X(I)+S
```

сокращает время выполнения цикла на время, необходимое для выполнения 1000 умножений.

4) Разбиение цикла противоположно объединению. Разбиение целесообразно делать, если внутри цикла есть условный оператор с проверяемым условием, которое не изменяется в цикле. Например,

```
      DO 1 I=1,100
      IF (A.GT.0.) X(I)=Y(I)
      X(I)=0.
1     CONTINUE
```

Условный оператор IF выполняется 100 раз, хотя можно преобразовать программу так, что он будет выполняться только один раз

```
      IF (A.GT.0.) GO TO 1
      GO TO 3
1     DO 2 I=1,100
2     X(I)=Y(I)
      GO TO 5
3     DO 4 I=1,100
4     X(I)=0.
5     CONTINUE
```

Время выполнения этого фрагмента уменьшается за счет увеличения необходимой памяти.

5) Правильное вложение циклов может увеличить быстродействие программы за счет уменьшения числа операций инициирования цикла. В следующем примере

```
      DO 1 I=1,100;          DO 1 I=1,10
      DO 1 J=1,10;           DO 1 J=1,100
1     A(I,J)=0.;           1 A(J,I)=0.
```

левый внутренний цикл инициируется 100 раз, а в правом — 10 раз.

Наконец, следует придерживаться простого правила: короткие циклы применять нецелесообразно, так как затраты на их организацию сравнимы с выполнением операторов цикла.

3.4.8. Ускорение на передачах управления. Различные операторы передачи управления:

безусловный оператор GO TO,
логический оператор IF,
арифметический оператор IF

— отличаются по своему быстродействию. Выше приведен список в порядке убывания быстродействия. Однако для некоторых ЭВМ последние два оператора меняются местами. Кроме того, для некоторых ЭВМ в условных операторах перехода сравнение с нулем выполняется значительно быстрее, чем с другим выражением. Это связано с использованием какой-либо быстрой команды. Поэтому оператор

```
IF (I.GT.J) X=1.
```

будет выполняться медленнее следующего оператора

```
IF (I-J.GT.0) X=1.
```

Ускорение на передаче управления можно получить, разбивая логическое выражение на составляющие, в том случае, когда результат логического выражения может быть уже известен, но оно продолжает вычисляться. Например, оператор

```
IF (A.GT.B.OR.C.LT.D) GO TO 1
```

разложенный на два оператора

```
IF (A.GT.B) GO TO 1
IF (C.LT.D) GO TO 1
```

в том случае, когда A>B, не требует выполнения второго оператора.

3.4.9. Ускорение передачи аргументов в функции-подпрограммы и подпрограммы. Фортран содержит два основных способа обмена информацией между подпрограммами: через COMMON-блоки и через аргументы. Ниже приведены два указанных варианта:

```
COMMON /BLOCK/A,B,C,D;          DATA A,B,C/0.,1.,2./
DATA A,B,C/0.,1.,2./;           CALL S(A,B,C,X)
CALL S(X);                      Y=SIN(X)
Y=SIN(X);
```

В зависимости от трансляции эти два способа передачи могут сильно отличаться по затратам времени и требуемой памяти. Как правило, первый способ предпочтительнее второго, поэтому целесообразно минимизировать число элементов в списке аргументов, размещая часть аргументов в COMMON-блоки.

Причина преимущества передачи аргументов в первом варианте заключается в том, что в вызывающей программе сначала накапливаются требуемые аргументы для замены формальных параметров (так же как и COMMON-переменных), но затем необходимо выполнить дополнительную работу, чтобы построить таблицу адресов аргументов и указатель на эту таблицу. В вызываемой программе таблица восстанавливается для определения места аргументов.

3.4.10. Оптимизация ввода — вывода. Обратим внимание на следующие аспекты ввода — вывода, которые повышают эффективность этих операций.

1) Простота списков ввода — вывода. Каждый элемент списка требует обращения к программе ввода — вывода библиотеки фортрана, которая, в свою очередь, вызывает системные программы. Поэтому, например, вывод на печать с помощью операторов

```
REAL A(100),B(100);                REAL A(100),B(100)
WRITE (6,*) A,B;                   WRITE (6,*)(A(I),B(I),I=
                                   =1,100)
```

в варианте слева потребует двух обращений к программе вывода, в варианте справа — 200 обращений.

2) Учет формы хранения массивов в памяти. Эффективность ввода — вывода массивов резко падает, если список не соответствует принятому способу хранения массивов в памяти на фортране. Например, вывод

```
WRITE (6,*) ((A(I,J),J=1,20),I=1,20)
```

будет менее эффективным, нежели

```
WRITE (6,*) ((A(I,J),I=1,20),J=1,20)
```

поскольку в памяти двумерные массивы запоминаются по столбцам, а не по строкам.

3) Бесформатный ввод — вывод промежуточных результатов. Бесформатный ввод — вывод определяется в **3.5.** Он служит для записи и чтения данных на внешней памяти без всяких преобразований (во внутреннем машинном представлении). Форматный ввод — вывод нужен только для пользователя, но не для программы. Так как преобразования в соответствие с форматом сопряжены со значительным числом операций и с увеличением необходимой памяти, то от него следует отказаться, если данные имеют смысл промежуточных.

Например, для хранения вещественного числа без формата требуется 4 байт, с форматом E15.7 необходимо 15 байт.

Наконец, при форматных преобразованиях может теряться точность представления чисел, а этого всегда следует избегать.

4) Экономия бумаги. Если при работе на дисплее возможен вывод по одному значению в строке, то такая манера выдачи на бумагу АЦПУ достойна осуждения, поскольку приведет к большому расходу бумаги. Следует познакомиться с числом позиций в строке АЦПУ, доступных для вывода (обычно 128 или 132), а затем, используя список вывода и оператор FORMAT, так расположить информацию, чтобы максимально использовалась вся доступная площадь бумаги.

● 3.5. Расширение возможностей фортрана

3.5.1. Введение. В **3.2** рассматривались некоторые конструкции языка фортран, которые в основном соответствуют стандарту фортрана 66.

Особенности версии фортрана 77 с расширенными возможностями (по сравнению с фортраном 66) приведены в Приложении 2.

Перечислим новые конструкции языка фортран 77, отсутствовавшие в фортране 66. Список новых элементов следующий:

PROGRAM;	ENTRY
CHARACTER;	SAVE
IMPLICIT;	OPEN
PARAMETER;	CLOSE
INTRINSIC;	INQUIRE
PRINT *;	.EQV.
READ *;	.NEQV.

структурный логический оператор

```
IF ... THEN
...
  ELSE ... THEN
  ...
  ELSE ...
ENDIF
```

и управление вводом — выводом файла.

Таким образом, в **3.2** рассмотрены не все элементы и фортрана 66. Если при решении задачи (чтении чужих программ) окажется необходимым освоить элементы, не рассмотренные в настоящей книге (см. Приложение 2), то следует обратиться к [13].

Из новых элементов фортрана 77 отметим три наиболее существенных, которые принципиально расширили возможности фортрана 66.

Это символьный тип данных CHARACTER и операции с ним. Символьный тип данных наиболее часто употребляется в программах обработки текстов, создании текстовых процессоров. В программах научно-технических расчетов символьный тип данных обычно используется в комментариях, для которых вполне достаточно средств фортрана 66. Поэтому этот элемент фортрана 77 здесь не рассматривается.

Следующий принципиально новый элемент (который часто называют основным отличием этой версии фортрана) — это структурный логический оператор. Неудобство старого логического оператора состоит в том, что после логического выражения

IF (лог. выр.) выполняемый оператор

может стоять лишь один выполняемый оператор, что влечет за собой частое применение оператора GO TO. Программа с большим количеством операторов GO TO, как правило, плохо читаема и не удовлетворяет современным требованиям к стилю программирования. Структурный логический оператор значительно уменьшил частоту употребления в программах оператора GO TO.

Наконец, третьим, существенно новым, элементом являются средства управления вводом — выводом файла. В 50—60-е годы основными носителями информации для ввода данных и программ были перфокарты, вывод результатов производился на АЦПУ или на перфокарты. Это наложило отпечаток на конструкции ввода — вывода фортрана 66. В настоящее время перфокарты

практически вышли из употребления, используются магнитные носители — ленты, диски разнообразной конструкции. Ввод программ и данных производится с терминалов в файлы на магнитных носителях, непосредственно связанных с ЭВМ. Вывод результатов может быть произведен также в файлы, либо на терминалы. Такой ввод — вывод потребовал создания гибкой системы управления вводом — выводом файла, что и нашло свое отражение в фортране 77.

3.5.2. Структурный логический оператор. Структурный логический оператор имеет следующий вид:

```
IF (логическое выражение) THEN
...
ENDIF
```

На месте многоточия может находиться любое число операторов, которые должны выполняться, если логическое выражение истинно. Если выражение ложно, то управление передается на следующий за ENDIF оператор.

Пусть, например, требуется найти максимальный элемент последовательности x_1, x_2, ..., x_{100} и определить его номер в последовательности. Фрагмент программы, решающей эту задачу, может быть таким:

```
      IM=1
      XM=X(1)
      DO  1  I=2,100
      IF  (XM.LT.X(I))  THEN
      IM=I
      XM=X(I)
      ENDIF
1     CONTINUE
```

Следующей конструкцией структурного логического оператора является

```
IF (логическое выражение) THEN
1-я группа операторов
...
ELSE
2-я группа операторов
...
ENDIF
```

Первая группа операторов выполняется, если логическое выражение истинно, вторая группа — если ложно. Эта конструкция полностью отвечает 2-й базовой логической схеме (см. гл. 2).

Например, вычисление функции

$$y=f(x)=\begin{cases} e^{-x^2}, & x\geqslant 0, \\ e^{x}, & x<0, \end{cases}$$

можно оформить с помощью этой конструкции так:

```
REAL FUNCTION Y(X)
REAL X
IF (X.GE.0.) THEN
Y=EXP(-X*X)
   ELSE
   Y=EXP(X)
ENDIF
RETURN
END
```

«Хорошим стилем» в оформлении программ является сдвиг операторов, принадлежащих своему IF или ELSE, как это сделано выше в примере.

Наконец, последней конструкцией являются вложенные логические операторы. Простейшей такой конструкцией является

```
IF (логическое выражение) THEN
...
ELSE IF (логическое выражение) THEN
...
ELSE
...
ENDIF
...
ENDIF
```

Заметим, что каждый оператор IF должен иметь закрывающий его оператор END IF, а после операторов IF, ELSE, ELSE IF могут отсутствовать операторы.

Например, вычисление функции

$$y = f(x) = \begin{cases} e^x, & x \geqslant 2, \\ \sin x, & 0 \leqslant x < 2, \\ \cos x, & x < 0, \end{cases}$$

можно оформить с помощью этой конструкции так:

```
REAL FUNCTION Y(X)
REAL X
IF (X.LT.2) THEN
ELSE IF(X.GE.0) THEN
Y=SIN(X)
ELSE
Y=COS(X)
ENDIF
ELSE
Y=EXP(X)
ENDIF
RETURN
END
```

Структурные логические операторы, образующие один уровень вложения, сдвигаются вправо на одинаковое число позиций. Например, если имеется три уровня вложения, то следует операторы представлять в форме

```
IF (L1) THEN
g1
    IF (L2) THEN
    g2
        IF (L3) THEN
        g3
            ELSE IF (L4) THEN
            g4
               ELSE
               g5
            ENDIF
        ELSE
        g6
        ENDIF
    ELSE
    g7
    ENDIF
ENDIF
```

Заметим, что приведенная структура операторов эквивалентна логической схеме рис. 3.2, где L — логические выражения, g — операторы, истине логических выражений соответствует: да. Если, например, логическое выражение L1 ложно, то выполняется группа операторов g7 и затем происходит выход из всей схемы.

На основании приведенного примера можно заключить, что на фортране 77 можно написать достаточно сложные логические схемы, используя удобные структурные логические операторы.

В качестве основного ограничения использования вложенных логических операторов выступает следующее правило: недопустима передача управления извне IF или ELSE оператору, находящемуся внутри, т. е. без проверки логического выражения.

Ни один из операторов группы g1 не может быть оператором GO TO, передающим управление какому-либо оператору группы g2.

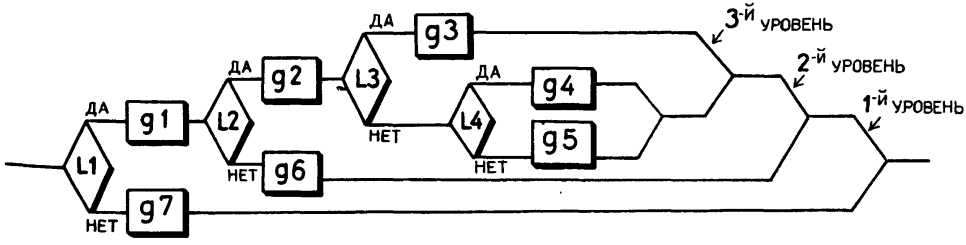

Рис. 3.2

3.5.3. Расширение возможностей ввода — вывода. Единицей ввода — вывода в фортране 77 является запись. Физическое представление записи зависит от конкретной ЭВМ. Записью может быть строка печати или строка файла на магнитном диске.

Записи делятся на форматные и бесформатные. Последняя запись файла называется записью конца файла. Файлу можно дать имя, связать с определенным каналом ввода — вывода с помощью логического номера устройства. Запись в файл можно ввести, вывести в двух режимах: в последовательном доступе, в прямом доступе. В последовательном доступе N записей переписывается в файл последовательно начиная с 1-й: 1, 2, ..., N, записи считываются также последовательно в том же порядке 1, 2, ..., N. В прямом доступе можно ввести или вывести любую запись, указав номер n, $1 \leqslant n \leqslant N$.

Файл имеет указатель, который при первом обращении к нему или при открытии файла устаналивается на запись номер 1. При каждом следующем обращении указатель передвигается по записям до встречи записи конца файла.

В последовательном файле записи должны быть либо только форматные, либо только бесформатные, длина записей может быть различной. В файле прямого доступа записи должны быть еще и одинаковой длины.

Обращение к файлам последовательного и прямого доступа — различное.

Общий вид операторов READ и WRITE в фортране 77 следующий:

READ (⟨управляющий список⟩) ⟨список ввода⟩,
WRITE (⟨управляющий список⟩) ⟨список вывода⟩,

где управляющий список может состоять из элементов

UNIT=, FMT=, REC=, END=, ERR=, IOSTAT=

или в переводе на русский язык

устройство=, формат=, запись=, конец файла=, ошибка=, статус вв-вы=;

в правой части равенства помещаются элементы согласно Приложению 2.

Частным случаем ввода, вывода форматной записи является рассмотренный в **3.2.** пример

```
      READ (UNIT=5, FMT=1) X, Y
1     FORMAT (2F6.3)
      WRITE (UNIT=6, FMT=1) A, B, C
```

где опускались лёвые части равенств управляющего списка

```
      READ (5, 1) X, Y
1     FORMAT (2F6.3)
      WRITE (6, 1) A, B, C
```

Эта форма правильна и в фортране 77, но если в управляющем списке отсутствует UNIT=, то номер устройства должен стоять

на первом месте, если отсутствует FMT=, то метка формата должна быть на втором месте списка. Например,

READ (5, 1, REC=15, IOSTAT=I, ERR=10, END=20) X

означает ввод на устройстве 5 по формату с меткой 1 записи 15, в случае ошибки переменной I будет присвоено положительное значение, равное индексу ошибки ввода, или отрицательное, если обнаружен конец файла, при безошибочном вводе или если не найден конец файла переменная I будет равна нулю.

Меткой 10 помечен оператор, на который передается управление в случае ошибки, меткой 20 — оператор, которому передается управление, если обнаружен конец файла.

Если ввод осуществляется из бесформатного файла, то простейшая форма оператора READ (без контроля ошибок ввода и конца файла) может быть такой:

READ (5) A

Здесь 5 — логический номер устройства, с которого будет считана очередная запись для элемента A.

При бесформатном вводе — выводе экономится время выполнения этих операций за счет отказа от преобразования данных, обеспечивается большая точность в выведении данных и, как правило, экономится место, требующееся для хранения файла. Бесформатный ввод — вывод обычно используется, когда вывод данных из некоторой программы впоследствии должен быть вводом в эту или другую программу.

3.5.4. Операторы OPEN, CLOSE. Прежде чем файл окажется доступным для операторов ввода — вывода (READ, WRITE) на логическом устройстве, файл должен быть создан и связан с этим устройством. Обычно это можно сделать вне фортран-программ, в рамках операционной системы: устройство 5 связывается с вводом — выводом на дисплее, 6 — с АЦПУ. Поэтому операторы READ, WRITE будут передавать данные по этим каналам (5, 6) без каких-либо усилий со стороны пользователя.

Однако если требуется несколько каналов ввода — вывода в программе, то необходимо уметь управлять связью файлов из самой программы. Это, например, делают, чтобы сохранить промежуточные результаты программы в файле данных для дальнейшего продолжения вычислений.

Оператор OPEN (открыть) создает новый файл и связывает его с устройством либо связывает с устройством уже существующий файл. В любой момент один файл должен быть связан только с одним каналом.

К концу выполнения программы все открытые файлы программа должна сама закрыть (хотя закрыть может и операционная система, но это плохой стиль программирования).

Оператор OPEN имеет следующий вид:

OPEN (⟨список открытия⟩)

Список открытия может состоять из элементов

UNIT=, FILE=, STATUS=, FORM=, BLANK=, ERR=, IOSTAT;

здесь в правой части равенств помещаются элементы согласно приложению 2, определяющие список открытия. Например,

UNIT=8	— номер устройства,
FILE='INTEGRAL'	— имя файла, который должен быть связан с устройством 8; если файла не существует, то он создается;
STATUS='OLD'	— файл существует; если не существует — указать «NEW»; если создается временный с удалением после закрытия — указать «SCRATCH»;
ACCESS='SEQUENTIAL'	— файл последовательного доступа; если прямого — указать «DIRECT»;
FORM='FORMATTED'	— доступ с форматным вводом — выводом; если с бесформатным — указать «UNFORMATTED»;
RECL=80	— длина записи в символах для файлов прямого доступа, для последовательного доступа описатель опустить;
BLANK='ZERO'	— все пробелы в числовых полях будут интерпретироваться нулями; если нужно, чтобы пробелы игнорировались, указать «NULL»
ERR=10, IOSTAT=I	— имеют тот же смысл, что в п. 3.5.3.

Этот файл может быть прочитан с помощью форматных операторов ввода — вывода, например таких:

```
      READ (8, 1) (A,(J), J=1, 16)
1     FORMAT (F5.3)
```

Файл отсоединяется от устройства с помощью оператора CLOSE. Оператор имеет следующий вид:

CLOSE (⟨список закрытия⟩)

Список закрытия может содержать следующие элементы:

UNIT=, IOSTAT=, ERR=, STATUS=

Все элементы, кроме последнего, уже были определены.

STATUS='KEEP' — файл сохраняется после закрытия; «DELETE», если файл должен быть удален; если этот элемент опущен, то будет использован «KEEP».

Например, закрытие файла (без контроля ошибок) с сохранением на устройстве 8 можно выполнить оператором

CLOSE (8)

3.5.5. Вычисления с двойной точностью. Для оценки вычислительной погрешности бывает необходимо преобразовать программу, выполняющую вычисления с одинарной точностью, в программу, вычисляющую с двойной точностью. Для этого следует:

1) преобразовать все вещественные константы в константы с двойной точностью, например

1.E−6→1.D−6

2) Описать все вещественные переменные, массивы как переменные двойной точности, например

REAL A, X(8)→DOUBLE PRECISION A, X(8)

3) преобразовать спецификации форматов ввода — вывода к двойной точности; данные двойной точности вводятся и выводятся с помощью спецификации формата

D*m.n*

где *m* — общее число позиций, *n* — число цифр после десятичной точки. Эта спецификация аналогична формату E*m.n* с учетом того, что теперь увеличивается возможное количество знаков после десятичной точки до 17; например, возможно задать такой формат

D20.13

4) Использовать во всей программе функции двойной точности, для этого библиотечные функции необходимо заменить на стандартные функции двойной точности, такие, как

DSQRT, DEXP, DCOS

а функции-подпрограммы употреблять с описанием типа, например

DOUBLE PRECISION SMALL(A, B)

3.5.6. Обобщение описания размерности. В фортране 77 размерность можно описывать задавая верхнюю и нижнюю границы. Например, для одномерного массива A можно описать размерность в форме

DIMENSION A(n_1: n_2)

где n_1, n_2 могут быть целыми арифметическими выражениями. Если n_1 опускается, то, как и в фортране 66, нижней границей считается 1.

Так как n_1, n_2 — выражения, то они могут принимать отрицательные, нулевые и положительные значения: должно быть только $n_2 > n_1$.

Например, оператор

```
DIMENSION A(-2:2)
```

описывает массив из пяти элементов

A(−2), A(−1), A(0), A(1), A(2),

а оператор

```
DIMENSION A(0:2*N+1)
```

при N=2 описывает массив из шести элементов

A(0), A(1), A(2), A(3), A(4), A(5)

Последний пример указывает, что такое расширение описания удобно в алгоритмах вычислительной математики, где часто нумерацию элементов массива начинают с нуля.

Фортран 77 допускает употребление семимерных массивов в отличие от трехмерных фортрана 66.

3.5.7. Оператор цикла. Запись оператора цикла DO следующая:

DO n, $i=m_1, m_2, m_3$

здесь n — метка последнего оператора цикла, i — переменная целого, вещественного типа или двойной точности; m_1, m_2, m_3 — арифметические выражения над величинами целыми, вещественными или двойной точности. Например, возможен такой оператор цикла

```
DO 2, X=X0, X1, -H
```

Еще одним важным отличием оператора цикла фортрана 77 является то, что цикл может ни разу не выполняться. В фортране 66 цикл по крайней мере выполняется один раз, так как проверка значения переменной i цикла производится в конце цикла (не превышает ли i значение m_2), поэтому в фортране 66 цикл

```
DO 2 I=1, 0
```

один раз выполняется. В фортране 77 такая проверка осуществляется перед выполнением цикла, а выход из цикла происходит, если $i>m_2$.

Кроме того, переменная i сохраняет свое значение после выхода из цикла и может использоваться в следующих операторах. Таким образом, операторы

```
      DO 2 I=1, 100
      ...
2     CONTINUE
      M=I
```

приведут к тому, что переменной M будет присвоено значение M=101.

Запятая в операторе после метки n не является обязательной, но она оберегает пользователя от некоторых возможных ошибок, которые не просто обнаружить.

Глава 4

СИСТЕМНЫЕ И ИНСТРУМЕНТАЛЬНЫЕ ПРОГРАММЫ

● 4.1. Введение

4.1.1. Три типа программ. Все используемые на ЭВМ программы могут быть разделены на три типа:

1) прикладные программы;
2) системные программы;
3) инструментальные программы.

Прикладные программы разрабатываются для решения на ЭВМ конкретных научно-технических задач, как правило, специалистами той предметной области, где используются результаты вычислений. Например, прикладные программы решают задачи тепломассопереноса в реакторах АЭС, устойчивости системы управления летательных аппаратов, надежности строительных конструкций, управления объектами в реальном масштабе времени и т. п. Прикладные программы создаются на различных языках программирования, в частности на фортране.

Разработка прикладного программного обеспечения решения задач — наиболее массовый вид программирования, которым заняты большинство специалистов — непрофессионалов в программировании.

Два других типа программного обеспечения (системные и инструментальные программы) разрабатываются профессиональными программистами.

Системные программы выполняются вместе с прикладными программами и служат для управления ресурсами ЭВМ — центральным процессором, памятью, вводом — выводом. Это программы общего пользования, которые предназначены для всех пользователей ЭВМ. Системное программное обеспечение разрабатывается так, чтобы ЭВМ эффективно могла выполнять различные прикладные программы. Оно состоит из двух больших групп программ:

1) *операционные системы* (ОС);
2) *системы управления базами данных* (СУБД).

Вторую группу системных программ (СУБД) мы не рассматриваем; основные задачи ОС излагаются ниже.

Инструментальные программы — это программы, которые не участвуют на этапе выполнения прикладной программы, но служат для разработки прикладных (и системных) программ. К инструментальным программам, например, относятся:

1) редакторы;
2) трансляторы;
3) программа-библиотекарь;
4) отладочные программы;
5) графические пакеты программ;
6) построители блок-схем.

Наиболее широко из инструментальных программ используются редакторы и трансляторы с языков фортран, паскаль и т. п.

4.1.2. Операционная система. Название большого набора программ — *операционная система* — связано с тем, что до появления таких программ соответствующую работу выполняли операторы ЭВМ. Перечислим основные задачи, решаемые ОС.

1. Управление ресурсами ЭВМ: регистрация и реакция на аппаратные сбои, распределение заданий разных пользователей, реакция на прерывания; составление расписаний работ; распределение памяти, внешних устройств.

2. Дополнение к прикладным программам: загрузка прикладных программ; связь различных частей программ; управление вводом — выводом.

3. Управление хранением данных и программ: хранение данных и программ, не зависимое от физических устройств.

4. Управление связью между программами, выполняемыми на различных вычислительных устройствах.

5. Взаимодействие пользователя и ОС через терминалы (алфавитно-цифровые дисплеи, графические дисплеи и т. п.).

6. Защита программ и данных от несанкционированного доступа.

7. Диагностика аппаратуры, данных и программ, восстановление, дублирование программ, данных и аппаратуры.

Следует заметить, что ОС выполняют фактически коллективное обслуживание различных пользователей с основной целью — максимально загрузить ЭВМ. Эта цель, как правило, вступает в противоречие с целью пользователя, который предпочитает монопольно владеть всеми ресурсами машины. Поэтому следует ясно представлять, что за услуги ОС необходимо платить пользователю уменьшением оперативной памяти на величину, где размещается ядро ОС, уменьшением размера внешней памяти, где располагается остальная часть ОС. Очевидно, что уменьшается быстродействие прикладной программы, поскольку часть времени работы ЭВМ тратится на выполнение программ ОС. И все же эти издержки обходятся пользователю гораздо дешевле, чем разработка собственных программ, выполняющих некоторые функции ОС.

Пользователю ЭВМ важно знать характеристики различных ОС, которые могут работать на одной и той же аппаратуре, чтобы выбрать (если имеется такая возможность) наиболее подходящую для решения своих задач. Пользователь должен иметь достаточную информацию об ОС, которая применяется на машине, чтобы знать, почему, как и когда что-либо происходит в системе.

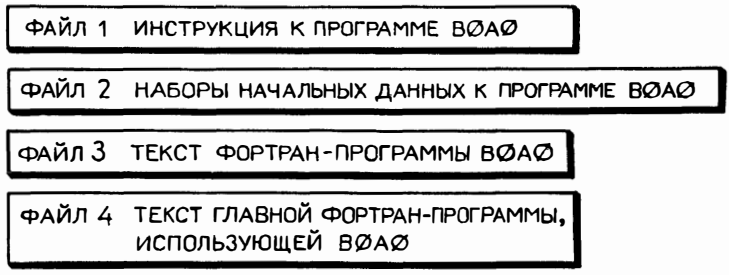

Рис. 4.1

4.1.3. Инструментальные программы. Рассмотрим программу редактор и программу-транслятор.

Программа *редактор* служит для формирования, исправления и хранения файлов, которые могут представлять собой наборы текстовой информации, наборы числовой информации, программы на алгоритмических языках (рис. 4.1). Процесс редактирования состоит из нескольких этапов:

1) открытие файла;
2) запись в файл;
3) редактирование (исправление) файла.

Если программа редактор готова принять команды редактирования файла, то она отводит рабочую область в оперативной памяти ЭВМ, называемую буфером. Информация, хранящаяся в буфере, не остается по окончании сеанса связи с ЭВМ. Для длительного хранения ее необходимо записать в файлы на дисках, лентах и т. п.

По командам пользователя можно открыть файл для редактирования — это входной файл. Информация по командам редактора поступает из входного файла в буфер редактирования, там обрабатывается и направляется в выходной файл (рис. 4.2). Есть специальные команды, по которым один и тот же файл объявляется как входным, так и выходным.

Существуют разнообразные редакторы, которые отличаются командами ввода информации, исправления ошибок (редактирования), но принципиального отличия они не имеют. Основные команды типичного редактора рассмотрены в **4.3**.

Программа *транслятор* — это вторая из инструментальных программ, с которой пользователь работает после редактора.

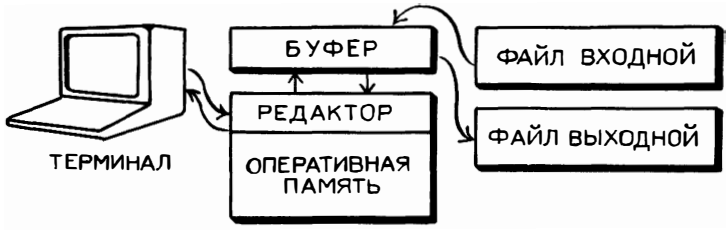

Рис. 4.2

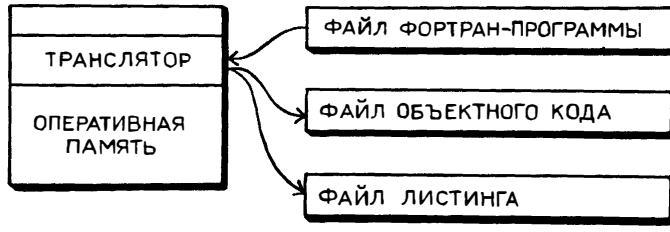

Рис. 4.3

Основной задачей транслятора является перевод программы, написанной на алгоритмическом языке, например фортране, в последовательность команд, почти готовых для выполнения ЭВМ. Оттранслированная программа называется объектным кодом.

Часто вместо транслятора употребляется инструментальная программа *компилятор*. Отличие программы компилятора от транслятора состоит в том, что компилятор кроме трансляции добавляет еще в программу нужные подпрограммы.

Транслирующая программа не может перевести исходную фортран-программу в форму, готовую для выполнения, хотя бы потому, что исходная программа может состоять из нескольких самостоятельных программных единиц, которые можно транслировать независимо. Следовательно, необходима еще программа — компоновщик или, что то же самое, *редактор связей*, построитель. Работа с типичным построителем описана в **4.3**.

Транслятор на входе имеет файл, содержащий текст фортран-программы, на выходе он может иметь два файла. В одном файле содержится объектный код, другой файл, называемый файлом листинга, содержит текст фортран-программы с указателями допущенных пользователем ошибок (рис. 4.3).

Транслятор обнаруживает как синтаксические ошибки (нарушение правил записи операторов), так и семантические (смысловые). К семантическим ошибкам относится, например, применение оператора GO TO n с отсутствующей меткой n в тексте программы.

После трансляции, если есть ошибки, обнаруженные транслятором, пользователь выводит на терминал или АЦПУ файл листинга. Затем следует возвратиться на этап редактирования, исправить ошибки и вновь уже исправленную фортран-программу транслировать. Весь описанный процесс: редактирование и трансляция — повторяется до тех пор, пока не будет получена безошибочная (с точки зрения транслятора) фортран-программа.

Таким образом, две инструментальные программы — редактор и транслятор — помогают пользователю подготовить прикладную программу для дальнейшего выполнения, т. е. проведения вычислений.

Не следует думать, что успешная трансляция программы — залог ее успешного выполнения. Программа будет выполняться с заданными входными числами, которые могут привести к ошибочным операторам, например, с точки зрения операций над числами

в ЭВМ: извлечение квадратного корня из отрицательного числа, выход за допустимый диапазон представления чисел (переполнение), логарифм отрицательного числа и т. п.

В этом случае необходимо тщательно проверить соответствие программы блок-схеме алгоритма. Поиски ошибок в программе и алгоритме составляет ту часть технологии вычислений, которую обычно называют отладкой.

Многие трансляторы предоставляют возможность специальным образом вводить отладочные операторы (например, операторы вывода), которые легко убрать из программы после отладки (см. **4.3**).

● 4.2. Элементы операционных систем

4.2.1. Введение. В настоящем пункте рассматриваются следующие вопросы: почему, что и как происходит в операционных системах. Изучаются только основные задачи ОС из тех, что перечислены в п. 4.1.2, а именно управление: памятью, процессором, устройствами, файлами, системой.

Изложение ведется на уровне общих понятий, для изучения деталей можно обратиться к специальной литературе [12, 14].

4.2.2. Управление памятью. Оперативная память — это тот ресурс ЭВМ, без которого нельзя выполнить программу. В любой момент времени, за исключением нескольких, все ячейки не используются. Они служат для хранения данных и программ одного или нескольких пользователей. Программы, которые ожидают выделения оперативной памяти, расположены во внешней памяти (например, на дисках). Часть оперативной памяти занята ядром ОС. *Ядро ОС* — набор управляющих программ, постоянно расположенных в оперативной памяти и обеспечивающих функционирование ОС.

Существует несколько стратегий распределения памяти, являющихся важной характеристикой ОС. Мы рассмотрим одну из наиболее простых стратегий, а именно распределение разделами с фиксированными границами.

Пусть вся оперативная память содержит 256 К байт, ядро ОС занимает 32 К. Тогда можно оставшуюся часть памяти разбить, например, на четыре раздела (рис. 4.4), которые содержат: 1 раздел — 32 К, 2—4 раздела — 64 К. С каждым разделом связывается очередь задач, которые соответствуют по размеру памяти раздела. Затем специальная программа — загрузчик — считывает программу задачи, программы библиотеки, необходимые для нее, и настраивает (устанавливает физические адреса) на раздел выполнения.

Программа, подлежащая загрузке и настройке, как раз и есть программа в объектном коде, которая получается в результате работы транслятора.

Режим работы, при котором в оперативной памяти может находиться несколько программ, называется *мультипрограммирова-*

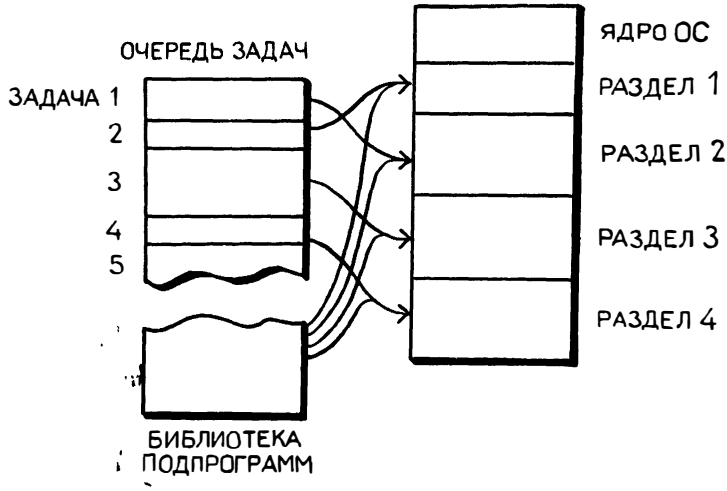

Рис. 4.4

нием. Основная цель мультипрограммирования — максимально загрузить процессор. Действительно, если некоторая программа перейдет в неактивное состояние (например, ждет ввода), то в памяти всегда есть программа, готовая к выполнению, а следовательно, процессор будет хорошо загружен.

Важной проблемой мультипрограммирования является защита системных программ, а также программ пользователей в своих разделах от возможных обращений к ним из соседних разделов, что приводит к порче программ и данных. Некоторые виды защиты реализуются аппаратно, некоторые выполняет ОС. Например, если из программы произойдет обращение к элементу массива, который при расчете его адреса оказывается в соседнем разделе, то ОС аварийно прерывает вычисления и выдает соответствующее сообщение.

4.2.3. Управление процессором. Процессор — ресурс ЭВМ, без которого нельзя выполнить программу. Выполнение программы называется процессом. Мультипрограммное использование одного процессора может создать иллюзию, что каждый процесс использует процессор независимо от остальных. Эту иллюзию создают программы управления процессором ОС, а именно: *планировщик задач*; диспетчер.

Планировщик выбирает из очереди задач программу и создает процесс, готовый к выполнению: выделяется оперативная память, внешняя память, файлы. Затем диспетчер управляет очередями готовых к выполнению процессов, выполняющегося процесса (только один) и заблокированных процессов, которые находятся в ожидании некоторого события (например, ввода или вывода). Наконец, планировщик завершает процесс после его полного выполнения.

Существуют различные процедуры организации работы диспетчера. Рассмотрим сравнительно простую и популярную цикличес-

кую процедуру. В ней каждому процессу по очереди выделяется одинаковый квант времени Δt (0,1—1 с), в конце которого, если процесс не завершился и не заблокирован, он снимается с процессора и ставится в конец очереди. В конец очереди ставятся появившиеся готовые и разблокированные процессы. Существенно влияет на управление очередью квант времени Δt и смесь процессов (короткие или длинные по времени выполнения).

Для совместного функционирования диспетчера, выполняющегося процесса и ввода—вывода в ОС организуются прерывания. *Прерывание* — это передача управления из выполняемого процесса на процесс обработки прерываний. Прерывание происходит по сигналу от таймера, когда истек квант времени Δt или по сигналу, что ввод—вывод завершен; это внешние прерывания. Прерывание может происходить из-за переполнения, деления на нуль, обращения к запрещенным ячейкам памяти т. п.

В момент прерывания аппаратура ЭВМ выполняет следующие действия:

1) в некоторую ячейку памяти заносится характеристика прерывания;

2) запоминается состояние прерванного процесса;

3) в счетчик команд заносится адрес, характерный для типа прерывания.

Затем выполняется программа обработки прерываний ОС, а также действия, соответствующие конкретному прерыванию, и возобновляется нормальная работа. В зависимости от типа прерываний процесс может быть продолжен, либо заблокирован, либо поставлен в конец очереди.

4.2.4. Управление устройствами. Управление работой устройств ЭВМ (алфавитно-цифровой дисплей, графический дисплей, магнитофон, АЦПУ и т. п.) осуществляется путем передачи им управляющих сигналов. Многие из устройств — электромеханические, не электронные и поэтому работают значительно медленнее, чем процессор, и, кроме того, асинхронно с ним. Эта несогласованность приводит к тому, что некоторое время процессор и устройство могут не быть в состоянии готовности обмениваться информацией. Операционная система должна сохранить в оперативной памяти данные, переданные процессором или устройством, но еще не полученные устройством или процессором. Такие области памяти называются буферами.

Чтобы согласовать работу быстрого процессора с медленными устройствами ввода—вывода (они должны работать параллельно с процессором), применяются аппаратные средства, называемые *каналами* или периферийными процессорами. Схема связи оперативной памяти (ОП) с устройством ввода—вывода (ВВ) через канал (К) показана на рис. 4.5, где УУ — устройство управления соответствующего ввода—вывода. Если канал могут попеременно использовать несколько медленных устройств, то он называется *мультиплексным*.

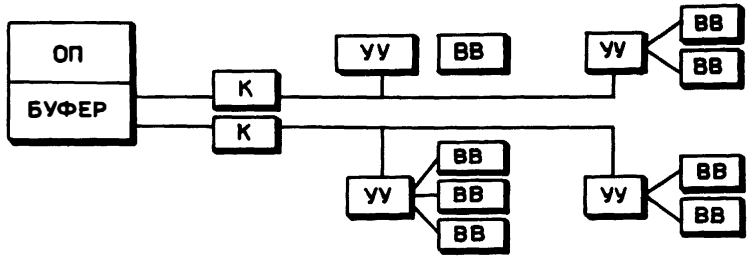

Рис. 4.5

Программы ОС, которые выполняют алгоритмы управления вводом — выводом, называются *драйверами*. Для каждого запроса ввода — вывода из выполняемой программы драйвер строит канальную программу или только ее часть, если тип устройства был известен транслятору (указали явно номер канала 6 в операторе WRITE (6, 1), 6→АЦПУ). Драйвер составляется для каждого типа устройства; эта программа обрабатывает прерывания возникающие, в частности, при ошибках ввода — вывода. Аппаратная часть системы ввода — вывода из-за присутствия механических деталей подвержена износу и приводит, как правило, к более частым сбоям в работе, чем электроника. Поэтому в драйвере предусматриваются повторные обмены, если обнаружена ошибка ввода — вывода, или прекращение процесса в случае неисправимой ошибки с предоставлением информации о виде ошибки.

Контроль правильности ввода — вывода осуществляется различными способами. Рассмотрим простейший контроль записи — чтения байта. К каждому байту добавляется информационный бит, который принимает значение 0 или 1 в зависимости от содержимого байта, но всегда так, чтобы сумма битов была нечетной для любого байта (рис. 4.6). С помощью суммирования проверяется четность каждого байта в процессе чтения. Нарушение четности свидетельствует об ошибке, а драйвер должен прервать ввод — вывод и отреагировать на ошибку.

4.2.5. Управление файлами. Система управления файлами является частью операционной системы, которая освобождает пользователя от трудностей, связанных с хранением файлов вне ЭВМ, а также предоставляет возможность нескольким пользователям обращаться к одному и тому же файлу. *Файловая система* обеспечивает: 1) независимость от процессора и внешнего устройства хранения файла, возможность обращения к файлу по символическому имени; 2) защиту информации от сбоев технических средств и программ, от несанкционированного доступа

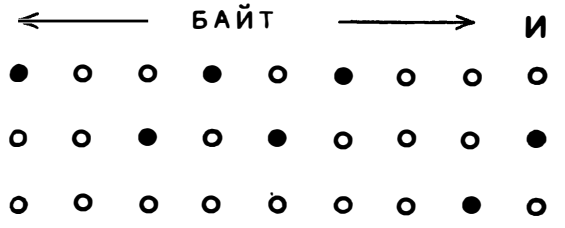

Рис. 4.6

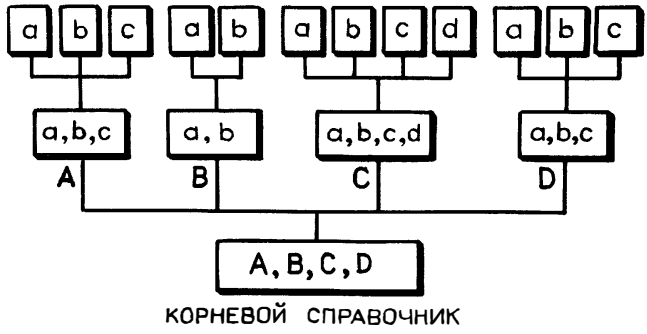

Рис. 4.7

к файлам; 3) простые средства общения пользователя с системой и эффективно распределяет внешнюю память под файлы.

Основные функции системы управления файлами делятся на два класса: пользовательские и автоматические.

По командам пользователя файл можно:

ПРОЧИТАТЬ; ЗАПИСАТЬ;
СОЗДАТЬ; УНИЧТОЖИТЬ;
КОПИРОВАТЬ; ПЕРЕИМЕНОВАТЬ.

Автоматические функции (без участия пользователя):

ПЕРЕМЕЩЕНИЕ ФАЙЛА

к которому редко обращается пользователь, во внешнюю (более медленную) память;

ДАМП ФАЙЛОВ

(сброс файлов на внешние носители для предотвращения потери информации);

ВЕДЕНИЕ КАТАЛОГА ФАЙЛОВ

состоит в организации системы файлов и хранении информации о них и их расположении в памяти.

Каждый файл в операционной системе должен иметь уникальное имя. Это имя присваивается системой путем добавления к имени файла пользователя некоторых атрибутов самого пользователя (его кода) либо добавлением времени создания файла (в однопроцессорных системах).

Доступ к информации в файле осуществляется в два этапа:

1) определение по имени его расположения;

2) поиск записи по ее позиции в файле.

Первый этап выполняется с помощью каталогов, которые могут быть организованы различными способами. Например, при древовидной организации каждая часть имени файла определяет ветвь, по которой следует перейти на следующий уровень дерева (рис. 4.7). Каждая вершина каталога является справочником соответствующего уровня, например, имя файла в двухуровневом

каталоге B; а на рис. 4.7. однозначно определяет путь по имени файла до его расположения.

Управление доступом к файлам осуществляется системой с помощью разбиения всех пользователей на классы. Обычно используются следующие четыре класса:

I. Владелец файла.
II. Друг владельца.
III. Назначенный пользователь.
IV. Рядовой пользователь.

Пользователи разных классов имеют разные права доступа к файлу. Типичный список прав:

1. Нет прав.
2. Знание о существовании файла.
3. Можно выполнить программу, нельзя скопировать.
4. Можно прочитать файл, выполнить программу, скопировать.
5. Можно добавить данные в конце файла.
6. Корректировка файла.
7. Изменение защиты файла.
8. Уничтожение файла.

Обычно каждое право включает все права, предшествующие в списке. Защита файла устанавливается указанием верхнего уровня прав для каждого класса. Можно установить, например, соответствие: I—8, II—5, III—4, IV—3. Некоторые операционные системы право доступа к файлам определяют паролями, которые не разглашаются пользователями.

4.2.6. Управление системой. Операционные системы берут на себя большую часть той работы, которую раньше выполнял оператор, но не всю. Соотношение между объемом, выполняемым на ЭВМ оператором, и операционной системой для разных систем сильно отличается.

В некоторых системах оператор включает ЭВМ, устанавливает носители информации, а остальное выполняет ОС; в других на оператора возлагается множество функций по поддержанию вычислительной системы.

В любом случае пользователь должен взаимодействовать либо с операционной системой, либо с оператором. Эти взаимодействия осуществляются на языке команд системы.

В диалоговом режиме запросы пользователя на языке команд обрабатываются при их появлении интерпретатором языка команд, иногда называемого оболочкой (в отличие от ядра). С помощью языка команд пользователь задает операционной системе программу действий. Такой язык может быть одновременно и языком программирования (как в системе UNIX). Большинство ОС предоставляют пользователю помощь по языку команд, которую можно получить командой HELP—«помоги».

Всюду в этом пункте предполагалось, что в памяти ЭВМ находится несколько программ, по крайней мере ядро ОС. Программы в память помещаются загрузчиком. Однако сам

загрузчик также когда-то должен быть загружен в оперативную память с помощью начального загрузчика. Эта процедура называется *загрузкой* ОС.

Потребность в загрузке ОС возникает: 1) после включения ЭВМ; 2) после обнаружения ошибок в работе ОС, которые не удается устранить другими способами. Число перезагрузок ОС в день определяет качество используемой системы.

Начальный загрузчик обычно реализуется аппаратно, вызывается нажатием клавишей в оперативную память и начинает выполнение программы вызова в память основного загрузчика. В функцию начальной загрузки входит очистка памяти («обнуление» содержимого всех ячеек).

4.3.7. Разработка операционных систем. Разработка операционных систем, как можно заключить из краткого описания решаемых ОС задач, довольно трудоемкое занятие, требующее высокой квалификации программиста. Небольшую ОС с ограниченными возможностями вполне по силам написать одному программисту. Для сравнения с ОС ЭВМ IBM-360, 370 приведем затраты на их создание, которые оцениваются несколькими миллиардами долларов в течение 25 лет.

Как правило, разработку ОС обеспечивают создатели серии ЭВМ, поскольку без ОС машина не найдет широкого применения у пользователей.

В следующем пункте рассматривается ОС реального времени, устаналиваемая на серию мини-ЭВМ. Зарубежным аналогом является ОС RSX 11M фирмы «DEC».

В последнее время получает все большую популярность операционная система UNIX (аналог ИНМОС), разработка которой не была связана с производителями ЭВМ. В ней нашли отражение последние достижения в теории программирования. Она может быть установлена на ЭВМ различных классов от микро- до супер-ЭВМ.

Возможно, что широкое распространение этой ОС в ближайшие годы потребует изучения именно этой системы уже на начальном этапе обучения. Подробнее познакомиться с ОС UNIX можно в [14].

● 4.3. Система реального времени

4.3.1. Введение. В этом разделе рассматриваются системные и инструментальные программы, которые обычно входят в операционную систему реального времени ОС РВ. Это одна из нескольких операционных систем, устанавливаемых на ЭВМ серии мини-машин СМ. Архитектура и состав технических средств таких машин описаны в гл. 1.

Как отмечалось в **4.2,** инструментальные программы не относятся к операционной системе. Однако, если понимать под системой реального времени ОС РВ объединение системных

и инструментальных программ, т. е. систему программирования (вычислительную среду) ЭВМ, то обосновано совместное рассмотрение этих программных средств в **4.3.**

Не все общие понятия, введенные в **4.1** и **4.2,** нашли свое отражение в этом разделе. Более подробно с системой программирования ОС РВ, а также с другой системой ОС РАФОС для мини-ЭВМ можно познакомиться в [5, 7].

4.3.2. Общие сведения о системе. Операционная система реального времени ОС РВ предназначена для работы с вычислительными комплексами на базе серии микро- и мини-ЭВМ: «Электроника 60», НЦ-80, ДВК1 — ДВК3, СМ 3, СМ 1300, «Электроника 100/16», СМ 4, СМ 1400, СМ 412, СМ 1420, СМ 1600, «Электроника 100/25», «Электроника 79».

Система рассчитана на работу с разнообразным оборудованием. Версия системы генерируется в зависимости от применения системы: от небольших систем для лабораторных исследований до больших систем обработки данных и управления.

Система ОС РВ ориентирована на магнитные диски и использует их как для сохранения системы и системных задач, так и в качестве основного носителя данных.

ОС РВ — мультипрограммная операционная система. Параллельное выполнение многих задач в режиме реального времени обеспечивается благодаря диспетчеризации, разбиению памяти на разделы, временной выгрузке задач на магнитный диск. Задача выполняется в разделе.

Система ОС РВ обеспечивает обслуживание многих терминалов, причем любой из них можно использовать в качестве командного терминала и вводить с него команды запуска, приостанова, отмены задачи.

Задачи в системе ОС РВ могут быть написаны на следующих языках программирования: макроассемблер, фортран, паскаль, бэйсик, кобол, си и др.

Минимальный комплект оборудования, необходимый для функционирования ОС РВ: центральный процессор, алфавитно-цифровой терминал, начальный аппаратный загрузчик, оперативная память (емкость 32 К байт или 48 К байт в зависимости от типа процессора), магнитный диск кассетного типа, таймер.

Максимальный размер оперативной памяти, например, для СМ 4 составляет 256 К байт, для «Электроники 79» — 4 М байт, из них 8 К байт отводится для адресации внешних устройств.

Система ОС РВ позволяет работать с гибкими дисками, перфоленточным устройством ввода — вывода, магнитной лентой, а также с другими устройствами из номенклатуры СМ ЭВМ.

4.3.3. Наименование внешних устройств. Во время генерации системы все внешние устройства, закрепленные за системой, полностью описываются. Каждому устройству дается двухбуквенное имя и обязательный одно- или двузначный восьмеричный номер устройства. (Например, DK3: — магнитный диск с номером 3). Если номер устройства опускается, система использует номер 0 по умолчанию.

Имена внешних устройств:
TT: — терминал,
LP: — АЦПУ,
MT: — магнитная лента,
DK: — магнитный диск кассетного типа,
PP: — устройство вывода на перфоленту,
PR: — устройство ввода с перфоленты,
DP: — пакетный магнитный диск.

Вводятся также псевдоустройства SY: и TI:

SY: — системное устройство. Обычно ему соответствует физическое устройство DK0:
TI: — терминал пользователя, за которым он работает.

4.3.4. Спецификация файла. Файл в ОС РВ, как и в любой файловой системе, имеет свое уникальное имя. Чтобы отличить файлы разных пользователей, вводится код идентификации пользователя (КИП), который состоит из двух восьмеричных чисел, разделенных запятой, каждое в диапазоне от 0 до 377.

Пример.

```
  1,15
234,3
 27,153
```

Каждый КИП связывается с каталогом (справочником) пользователя, который содержит имена файлов данного пользователя.

Любая системная программа в ОС РВ, которая обращается к файлам, делает это с помощью стандартной командной строки следующего вида:

ВЫВФ 1, ..., ВЫВФ N=ВВФ 1, ..., ВВФ N

где ВЫВФ — спецификация (характеристика) выводного файла, ВВФ — спецификация вводного файла.

Каждая спецификация файла (вводного или выводного) имеет следующий формат:

УСТР: [КИП] ИМЯ ФАЙЛА. ТИП; ВЕРСИЯ/КЛЮЧ 1 .../КЛЮЧ N

где УСТР: — физическое устройство, которое содержит файл, например DK1:

КИП — код идентификации пользователя.
Имя — может содержать до 9 буквенно-цифровых символов. Имя файла и тип всегда разделяются точкой.

Пример.

```
PROG 1
TEST
BIGMACCMD
```

ТИП — может содержать до трех буквенно-цифровых символов, является средством для различения файлов с одним и тем же именем. Например, программа на фортране может быть названа TEST.FTN, в то время как файл, содержащий объектный код этой программы, может быть назван TEST.OBJ. Тип файла и версия всегда разделяются точкой с запятой.

ВЕРСИЯ — восьмеричное число, используемое для обозначения различных версий файла. Когда файл впервые создается, ему приписывается номер версии, равный 1. Номер версии находится в диапазоне от 1 до 77777 (восьмеричное).

Пример.

```
TEXT.FIN;5
TEXT.OBJ;21
```

КЛЮЧ — двухбуквенное имя в символьном коде идентифицирующее тип ключа. Пусть ключ обозначается SW. Тогда SW устанавливает действие ключа, /—SW или /NOSW отменяют действие ключа. За ключом могут следовать значения.

Пример.

```
/TI:25:MAR:84
/CP
/LI:3
```

Примеры спецификаций файлов:

1) DK0: [230,15] TEXT.FTN;3

Третья версия файла TEXT.FTN находится на магнитном диске DK0: в каталоге 230,15.

2) MT3: [1,200] DAN158.BIN;1

Первая версия файла DAN158.BIN находится на магнитной ленте MT3: в каталоге 1,200.

Если какие-то элементы спецификации файла не указаны, то операционная система присваивает этим элементам определенные значения. Ниже слева указана отсутствующая спецификация, справа — значение, присвоенное системой.

УСТР — SY0: (системное устройство),
КИП — КИП, с которым пользователь вошел в систему,
ИМЯ ФАЙЛА — нет умолчания. Должно быть указано имя,
* (неявное имя),
ТИП — умолчание устанавливается в зависимости от вызванной системной задачи,
ВЕРСИЯ — для вводных файлов — самый последний номер версии. Для выводных файлов — последний номер версии +1,
/КЛЮЧ — умолчание устанавливается в зависимости от вызванной системной задачи.

Пример.

Пусть пользователь вошел в систему с КИП, равным 3,162, и с помощью программы EDI (редактор текста) создал файл с именем TEXT.FTN. Тогда использование спецификации TEXT.FTN эквивалентно следующему:

SY0: [3,162]TEXT.FTN;1

В спецификации имени файла часто используется символ *, который означает любой элемент. С помощью этого символа пользователь может одной командой задавать действие для нескольких файлов.

Примеры.

1) *.FTN означает все файлы с типом .FTN, такие, как TEXT.FTN, PROG.FTN и т. п.

2) PROG1.* означает все файлы с именем PROG1 и любым типом, такие, как PROG1.FTN, PROG1.OBJ и т. п.

3) PROG1.FTN; * означает все файлы PROG1.FTN с любой версией, такие, как PROG1.FTN;12, PROG1.FTN;3 и т. п.

4) [*, *]TEXT.*; * означает файлы во всех каталогах с именем TEXT с любым типом и любой версией.

4.3.5. Ввод с терминала. Имеется три типа отзыва, которые указывают, что терминал ожидает ввода:

1) Отзыв по умолчанию « > ».

2) Отзыв задачи «TSK > », где TSK — трехсимвольное имя задачи, например FOR > , TKB > .

3) Отзыв программы связи с оператором «MCR > ».

Программа связи с оператором (интерпретатор команд MCR) обслуживает команды пользователя, вводимые с терминала.

Терминал, готовый принять незапланированный ввод, выводит символ « > ». Понятие «незапланированный ввод» предполагает, что нет специальной задачи, требующей ввод с терминала. Ввод, следующий за отзывом по умолчанию, всегда неявно предназначен для программы связи с оператором «MCR». Ввод после отзыва «TSK» предназначен задаче, выдавшей этот отзыв. Например, транслятор с фортрана выдает отзыв FOR > .

Для того чтобы явно передать управление программе связи с оператором, необходимо ввести CTRL/C (одновременно нажать клавиши CTRL и C). Выдается приглашение MCR > , т. е. программа MCR требует ввода.

Ввод для программы MCR есть командная строка, которая содержит имя команды и требуемые параметры. Достаточно указать первые три буквы команды. Исключение составляет команда HELP. Имя команды отделяется от ее параметров пробелом. Параметрами команд MCR могут быть имя задачи, имя файла, спецификация устройства.

Примеры.

1) Вызвать транслятор с языка фортран

> FOR < CR >

Здесь < CR > — нажатие клавиши возврата каретки.

2) Распечатать на терминале текущие время и дату

> TIM < CR >

Для управления терминалом используются специальные символы. В табл. 4.1 приведены некоторые клавиши или комбинации клавиш терминала VDT (Видеотон). Ввод кода, как уже выше отмечалось, с помощью комбинации клавиш, например «CTRL/Q», производится путем одновременного нажатия двух клавиш: «CTRL» и «Q». В скобках указано наименование клавиш для терминала «Электроника».

4.3.6. Вход и выход в системе. Каждому пользователю, желающему работать в ОС РВ, администратор системы назначает КИП и пароль. Вход в систему осуществляется по команде HELLO программы MCR. Система запрашивает ввод КИП и пароля.

Таблица 4.1

Ввод с клавиатуры	Действие
CTRL/C (СУ/С)	Вызывает передачу управления программе MCR
TAB (ГТ)	Вызывает перемещение к следующей позиции табуляции
CTRL/O (СУ/О)	Используется для подавления и возобновления вывода на терминал
RETURN (ВК)	Вызывает окончание ввода строки и возврат каретки к началу строки
LINE FEED (ПС)	Вызывает перемещение к первой позиции на следующей строке
DEL (ЗБ)	Удаляет последний введенный символ
CTRL/S (СУ/Q)	CTRL/Q возобновляет вывод, приостановленный по CTRL/S
CTRL/S (СУ/S)	Во время вывода вызывает приостанов вывода. Можно продолжить вывод по CTRL/Q
CTRL/U (СУ/U)	Вызывает удаление вводимой строки до символа « > ». Сопровождается на экране как ┐U
CTRL/Z (СУ/Z)	Используется для завершения задачи. Сопровождается на экране ┐Z

Допустим, пользователю назначили КИП, равный 3,162, и пароль DIMA5. Тогда он может войти в систему следующим образом:

> HEL 3,162 < CR >
PASSWORD: DIMA5 < CR >

Текст, выданный системой, здесь и ниже подчеркнут. Во время ввода пароля символы не отображаются на экране терминала.

После регистрации системой входа на терминал будет выведено:

ОС РВ MULTI—USER SYSTEM
COOD MORNING
Дата время LOGGED ON TERMINAL имя терминала
>

Вывод подсказки « > » означает, что система готова к работе и ожидает ввод от пользователя.

После окончания работы выход из системы осуществляется по команде BYE программы MCR:

> BYE < CR >

На терминал выводится сообщение о завершении работы данного пользователя:

HAVE A GOOD MORNING
Дата время имя терминала LOGGED OFF
>

Для того чтобы снова начать работать в системе ОС РВ с данного терминала, необходимо задать команду HELLO и т. д.

Процесс регистрации входа и выхода позволяет системе хранить информацию о каждом пользователе.

4.3.7. Подготовка программы на фортране для выполнения. Процесс подготовки программы можно представить в виде схемы рис. 4.8. Из схемы видно, что процесс подготовки состоит из нескольких шагов:

1) С помощью редактора текстов создается файл, содержащий исходный текст программы на фортране.

2) Транслятор с языка фортран преобразует исходный файл в файл в объектном коде.

3) Построитель задач преобразует файл в объектном коде в файл образа задачи, готовый к выполнению.

4) По команде RUN программы MCR файл образа задачи запускается на выполнение.

Для иллюстрации схемы прохождения фортран-программы рассмотрим программу вычисления значения функции

$$f(x)=x^2+0{,}01\sin(x)$$

в точке x_0, значения x_0 подлежит вводить с терминала, $f(x_0)$ — выводить на терминал.

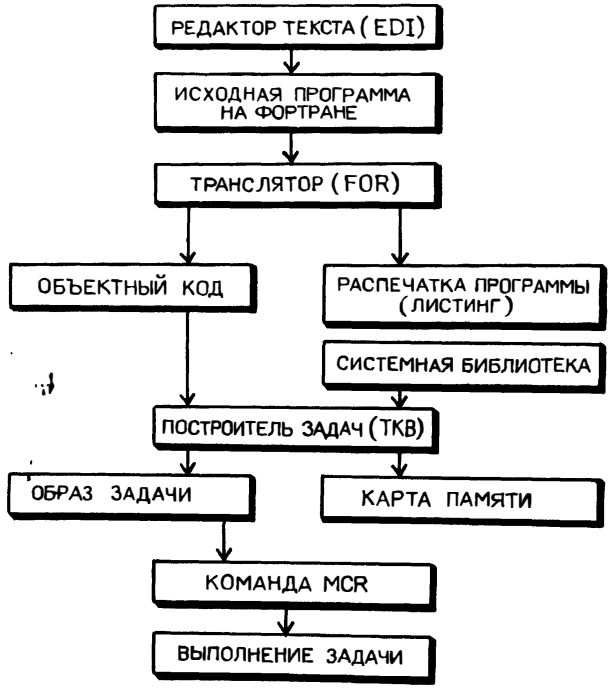

Рис. 4.8

Этап 1. Вход в систему. Схема прохождения фортран-программы состоит из следующих этапов:

```
> HEL 3,162 <CR>
PASSWORD DIMA5 <CR>
OC PB MULTI—USER SYSTEM
GOOD MORNING
>
```

Этап 2. Вызов программы «Редактор текста» (EDI), ввод текста программы:

```
        > EDI PROG.FTN <CR>
        CREATING NEW FILE
        PAGE 0
        INPUT
        REAL XO,FX <CR>
        READ (5,10)XO <CR>
10      FORMAT (E13.6) <CR>
        FX=XO**2+0.01*SIN(XO) <CR>
        WRITE (5,10) FX <CR>
        END <CR>
        <CR>
*EX   <CR>
EXIT
```

Этап 3. Вызов транслятора с фортрана (FOR):

> FOR PROG, LP: =PROG < CR >

Этап 4. Вызов построителя задач (ТКВ):

> ТКВ PROG=PROG < CR >

Этап 5. Запуск задачи на выполнение:

> RUN PROG < CR >

После запуска задачи PROG на выполнение произойдет прерывание и задача перейдет в ожидание запланированного ввода значения x_0 по заказанному формату E13.6. Пусть требуется вычислить значение $f(0,3)$. Тогда можно набрать на терминале

3.0E−01 < CR >

После нажатия клавиши возврата каретки задача PROG выполняется и на терминал будет выведено значение $f(0,3)$.

Чтобы вычислить значение $f(0,5)$, потребуется новый запуск задачи на выполнение

> RUN PROG < CR >

ввод с терминала значения 0,5 и т. д.

Завершается работа в системе выходом из ОС РВ.

Этап 6. Выход из системы

> BYE < CR >
HAVE A GOOD MORNING

Дата время имя терминала LOGGED OFF

Рассмотренные этапы 1—6 представляют собой простейший дисплейный сеанс пользователя в ОС РВ.

4.3.8. Редактор. Текстовый редактор в режиме диалога позволяет создавать или корректировать исходные программы и другие текстовые материалы в символьном виде.

Инструментальная программа текстовый редактор вызывается командой EDI программы MCR:

> EDI < CR >
EDI >

В ответ на подсказку «EDI > » пользователь должен ввести спецификацию редактируемого файла, которая имеет вид

УСТР: [КИП] ИМЯ ФАЙЛА. ТИП

Пусть пользователь вошел в систему с КИП, равным 3,162, и системным устройством является DK0: . Тогда для создания в каталоге 3,162 на DK0: файла TEXT.FTN или для редактирования существующего файла с таким же именем в ответ на подсказку «EDI > » необходимо ввести следующую строку:

EDI > TEXT.FTN < CR >

Если создается новый файл, то редактор печатает

CREATING NEW FILE
INPUT

и переходит в режим ввода текста с терминала. Если файл с указанным именем существует, то текстовый редактор считывает в оперативную память (в блочный буфер) первый блок текста из файла (весь файл может состоять из одного блока), печатает

PAGE 0

и в режиме команд ждет ввода первой команды.

Редактор текста допускает два режима работы: режим команд и режим ввода. Переход из режима команд в режим ввода осуществляется командой INSERT. Все строки, вводимые с этого момента, будут добавлены к файлу как новый текст, следующий за текущей строкой. Для перехода из режима ввода в режим команд необходимо ввести символ < CR > как первый символ в строке. Текстовый редактор выводит при этом символ подсказки «*», который означает, что установлен режим команд.

Указатель текущей строки всегда указывает на первый символ в строке. После считывания в блочный буфер первого блока текста из редактируемого файла указатель текущей строки устанавливается на начало первой строки. Некоторые команды редактирования могут перемещать указатель текущей строки.

Редактор можно вызывать укороченной командой, после которой не будет подсказки «EDI», например

> EDI TEXT.FTN < CR >

Основные команды текстового редактора

Все команды вводятся в ответ на приглашение «*». Имя команды отделяется пробелом от ее параметров, каждая команда заканчивается символом < CR >.

Команда PRINT (напечатать)

Формат: P [N]

Команда распечатывает текущую строку и следующие за ней N−1 строк на терминале. Последняя распечатываемая строка становится текущей. Если N не указано, то значение N подразумевается равным 1. Ввод CR эквивалентно команде P 1.

Пример.

* P10 < CR >

Распечатать на терминале 10 строк начиная с текущей.

Команда DELET (удалить)

Формат: D [N]

Команда вызывает удаление текстовых строк следующим образом:

1) Если задается «+N», то текущая строка и N−1 строк, следующих за текущей, удаляются. Текущей строкой становится строка, следующая за последней удаленной строкой.

2) Если задается «−N», то удаляются N строк, предшествующих текущей строке. Указатель текущей строки остается неизменным.

Если N не указано, то значение подразумевается равным +1.

Пример.

* D12 < CR >

Удалить 12 строк начиная с текущей.

Команда LOCATE (определить место)

Формат: [N] L подстрока.

Команда исследует блочный буфер начиная со строки, следующей за текущей строкой, и ищет строку, содержащую заданную подстроку (подстрока — часть строки). Числовое значение N, предшествующее команде, запрашивает поиск N-го появления заданной подстроки. Указатель строки устанавливается на строке, содержащей указанную подстроку. Если N не указано, то значение N подразумевается равным +1.

Пример.

* L (AI.CT.X) < CR >

IF (AI.CT.X) GO TO 25

Команда CHANGE (заменить)

Формат: C/ПОДСТРОКА 1/ПОДСТРОКА 2

Команда осуществляет поиск подстроки 1 (часть строки 1) в текущей строке, и если находит ее, то заменяет на подстроку 2. Начальным и конечным ограничителем подстроки может быть любой символ, который не содержится в указанной подстроке, например, наклонная черта.

Пример.

Пусть исходный текст программы содержит строку

X=X−FI*ALPHA

тогда после выполнения команды

C/−FI/+BETA < CR >

получим следующий результат:

X=X+BETA*ALPHA

Команда LINE CHANGE (заменить строку)

Формат [N] L C/ПОДСТРОКА 1/ПОДСТРОКА 2

Команда заменяет подстроку 1 на подстроку 2 в текущей строке и N−1 строках за ней. Если N не указано, то значение N подразумевается равным +1.

Команда INSERT (вставить)

Формат: I

Команда устанавливает режим ввода. Может быть введен ряд новых строк, следующих за текущей строкой. Каждая вводимая строка заканчивается символом < CR >. Символ < CR > в качестве первого символа в строке вызывает возврат текстового редактора в режим команд и печать символа «*».

Пример.

Пусть исходная программа содержит строки

```
A=0.5*SIN(ALPHA)
B=X+Y+A*COS(BETA)
```

и указатель строки установлен на первой строке. Тогда после выполнения команды INSERT

```
*
I < CR >
X=SQRT(TETA) < CR >
Y=3.15/A+1. < CR >
< CR >
*
```

получим

```
A=0.5*SIN(ALPHA)
X=SQRT(TETA)
Y=3.15/A+1.
B=X+Y+A*COS(BETA)
```

Команда NEXT (следующий)

Формат: N [M]

Команда передвигает указатель строки на M строк вперед или назад от текущей строки. Если M не указано, значение M подразумевается равным +1.

Пример.

```
* N 15 < CR >
```

Переместить указатель строки на 15 строк вперед относительно текущей строки.

Команда OVERLAY (перекрыть)

Формат: O [N]

Команда вызывает удаление N строк и замену их на любое количество строк, введенных пользователем. После удаления строк текстовый редактор переходит в режим ввода (см. команду INSERT). Если N не указано, то значение N подразумевается равным +1.

Пример.

Пусть исходная программа содержит строки

```
A=1.
```

```
B=3.1415*R1
C=SIN(BETA)
```

и указатель строки установлен на 2-й строке. Тогда после выполнения команды

```
* O <CR>
D=8. <CR>
E=D*SIN(R1/R2) <CR>
<CR>
*
```

получим следующий текст:

```
A=1.
D=8.
E=D*SIN(R1/R2)
C=SIN(BETA)
```

Команда BEGIN (начинать)

Формат: B

Команда устанавливает указатель текущей строки на начало блочного буфера.

Пример.

```
* B <CR>
```

Команда READ (читать)

Формат: REA [N]

Команда позволяет считать следующие N блоков текста из файла в блочный буфер. Указанное число блоков не должно превышать емкость буфера. Если в буфере уже есть блоки, то новые блоки добавляются к ним. Если N не указано, то значение N подразумевается равным +1.

Пример.

```
* REA 3 <CR>
```

Считать три блока текста из файла в блочный буфер.

Команда WRITE (записать)

Формат: W

Команда записывает содержимое блочного буфера в выводной файл. Содержимое блочного буфера очищается.

Пример.

```
W <CR>
```

Команда RENEW (обновить)

Формат: REN [N]

Команда записывает текущий блочный буфер в выводной файл и считывает новый блок из выводного файла. Процесс повторяется N раз. Последний блок остается в блочном буфере. Если N не указано, то значение N подразумевается +1.

Команда EXIT (выход)

Формат: EX ВЫВ.Ф

Команда передает все оставшиеся строки из блочного буфера и вводного файла (в том же порядке) в выводной файл, закрывает файлы и завершает редактирование. Если указана спецификация файла, то выводной файл переименовывается в соответствии с ней. В том случае, если спецификация не указана и редактировался существующий файл, выводной файл создается с тем же именем, что и вводной, а номер версии увеличивается на 1.

Пример.

1) *EX PROG.FTN < CR >

Закончить редактирование и записать результат в файл с именем PROG.FTN

2) *EX < CR >

Закончить редактирование и записать результат в файл с тем же именем, что и вводной, с номером версии на единицу больше.

4.3.9. Транслятор с фортрана. Транслятор с фортрана представляет собой инструментальную программу, вызываемую командой FOR программы MCR. Он создает из исходного файла, содержащего текст программы, файл в объектном коде.

> FOR < CR >

FOR >

В ответ на подсказку «FOR > » пользователь должен ввести командную строку, которая имеет вид

FOR > объектный, файл =исходный < CR >
файл листинга файл

Транслятор можно вызвать укороченной командой

> FOR объектный, файл =исходный < CR >
файл листинга файл

Отличие этой команды и других аналогичных укороченных (в одну строку) команд состоит в том, что вызванные программы (FOR, ТКВ и т. п.) после выполнения работы удаляются из оперативной памяти. В то же время первый вариант команды оставляет транслятор в памяти и можно его использовать сразу же для трансляции других программных единиц.

Файл листинга содержит распечатку программы на фортране с указанием ошибок, которые были обнаружены в результате трансляции. Этот файл обычно выводится на терминал или АЦПУ.

Допускается указание одного или двух файлов вывода и только одного файла ввода.

Если явно не указан тип файла, то транслятор с фортрана ищет входные файлы с типом .FTN, а файлы в объектном коде создает с типом .OBJ.

Примеры.

Пусть пользователь работает в системе с КИП, равным 3,162, и файл TEXT.FTN содержит исходную программу.

1) FOR > TEXT,LP: =TEXT < CR >

В результате трансляции файла TEXT.FTN в каталоге 3,162 образуется файл TEXT.OBJ; листинг программ выдается на АЦПУ.

Данная командная строка эквивалентна следующей:

FOR > SY0: [3,162] TEXT.OBJ,LP: =SY0: [3,162] TEXT.FTN

2) FOR > TEXT=TEXT < CR >

Создается только файл TEXT.OBJ, а листинг не выводится.

3) FOR > ,LP: =TEXT < CR >

Файл в объектном коде не создается, а листинг выводится на АЦПУ.

Если заменить LP: на TI:, то листинг будет выводиться на терминал пользователя.

В случае появления ошибок при трансляции программы необходимо исправить их (с помощью редактора EDI) и повторить процесс трансляции снова. Это процесс продолжается до тех пор, пока файл в объектном коде не будет содержать ошибок. Теперь файл в объектном коде можно обрабатывать построителем задач.

Для выхода из транслятора с фортрана (передача управления программе MCR) необходимо ввести CTRL/Z (VDT) или СУ/Z («Электроника») в ответ на подсказку «FOR».

4.3.10. Построитель задач. Построитель задач представляет собой системную программу, вызываемую командой ТКВ программы MCR. Он объединяет файлы в объектном коде для построения файла образа задачи.

> ТКВ < CR >

ТКВ >

В ответ на подсказку «ТКВ > » пользователь должен ввести командную строку, которая имеет следующий вид:

ТКВ > файл образа задачи, файл карты памяти = объектный файл(ы) < CR >

(возможен также укороченный вариант команды).

Файл образа задачи содержит задачу, готовую к выполнению. Файл карты памяти содержит распределение оперативной памяти для данной задачи. Файл карты памяти обычно выводится на терминал или АЦПУ.

Допускается указание одного или двух файлов вывода и одного или нескольких файлов ввода.

Если явно не указан тип файла, то построитель задач ищет входные файлы с типом .OBJ, а файл образа задачи создает с типом .TSK.

Примеры.

Пусть пользователь работает в системе с КИП, равным 3,162, и файл TEXT.OBJ — файл, полученный в результате трансляции файла TEXT.FTN.

1) ТКВ > TEXT/CP,LP: =TEXT < CR >

В результате работы построителя задач из файла TEXT.OBJ в каталоге 3,162 образуется файл TEXT.TSK; карта памяти выводится на АЦПУ.

Данная команда эквивалентна следующей:

ТКВ > SY0: [3,162]TEXT.TSK/CP, LP: =
=SY0: [3,162]TEXT.OBJ < CR >

Ключ /CP означает, что задача выгружаемая, т. е. в процессе решения данной задачи система может выгрузить ее из оперативной памяти на магнитный диск, чтобы предоставить место задаче с более высоким приоритетом. Рекомендуется всегда указывать, что задача выгружаемая, если она таковой и является.

Для построения образа задачи из двух (и более) объектных файлов их имена перечисляются после знака равенства в командной строке, например

ТКВ > TEXT/CP,LP: =TEXT1,TEXT2,TEXT3 < CR >

В ОС РВ имеется FTB более быстрый, чем ТКВ, построитель задач, однако с ограниченными возможностями. Для применения этой программы следует всюду заменить ТКВ на FTB.

Для того чтобы построитель задач начал работу, после ввода командной строки в ответ на приглашение «ТКВ > » необходимо ввести две наклонные черты //, что означает «Конец входных данных».

Окончательный вид командных строк следующий:

ТКВ > TEXT/CP,LP: =TEXT < CR >
ТКВ > // < CR >
2)
ТКВ > TEXT/CP=TEXT < CR >
ТКВ // < CR >
Построение образа задачи; карта памяти не выводится.
3)
ТКВ > ,LP: =TEXT < CR >
ТКВ > // < CR >

Образ задачи не строится; карта памяти выводится на АЦПУ.

Для вывода карты памяти на терминал пользователя символы LP: необходимо заменить на TI:.

После окончания работы ТКВ управление передается программе MCR.

4.3.11. Программа работы с файлами. Программа выполняет следующие функции:

1) копирование файлов с одного устройства на другое;
2) удаление файлов;
3) распечатка каталога файлов пользователя и др.

Программа работы с файлами вызывается командой PIP программы MCR.

> PIP < CR >
PIP >

Возможен укороченный вариант команды.

В ответ на подсказку «PIP > » пользователь должен ввести командную строку для программы PIP. Формат командной строки может быть различным в зависимости от функций, которые выполняются по этой команде. Выход из программы PIP осуществляется вводом CTRL/Z.

Во всех командах, описанных ниже, «ВВФ» и «ВЫВФ» — спецификации вводных и выводных файлов вида:

УСТР: [КИП] ИМЯ ФАЙЛА.ТИП; ВЕРСИЯ/КЛЮЧ

Копирование файлов с одного устройства на другое

Формат командной строки:

ВЫВФ=ВВФ1,ВВФ2, ...,ВВФN

Примеры.

1) PIP > DX0:PRIM.DAT=DK2:TEXT.DAT < CR >

Копирование последней версии файла TEXT.DAT с DK2: в файл PRIM.DAT на гибком диске DX0:.

2) PIP > MT1:NEWT.FTN=DK0:NEWTON.FTN < CR >

Копирование последней версии файла NEWTON.FTN с диска DK0: в файл NEWT.FTN на магнитной ленте MT1:.

3) PIP > LP:=PROG.FTN < CR >

Печать на АЦПУ последней версии файла PROG.FTN.

4) PIP > TI:=*.LST < CR >

Печать на терминале последних версий всех файлов с типом .LST.

Удаление файлов

Формат командной строки:

ВВФ1,ВВФ2, ...,ВВФN/DE

В спецификациях файлов номер версии должен быть указан явно или через *.

Примеры.

1) PIP > DK2:TEST.DAT;5/DE < CR >

Удалить версию 5 файла TEST.DAT на DK2: из каталога по умолчанию.

2) PIP > *.OBJ; *,*.TMP; */DE < CR >

Удалить все версии всех файлов типа .OBJ и .TMP из каталога по умолчанию на устройстве по умолчанию.

Печать каталога

Формат командной строки

ВЫВФ=ВЫВФ1,ВЫВФ2, ...,ВВФN/LI

Если спецификация выводного файла не указана, то распечатка выводится на терминал пользователя. Распечатка каталога содержит следующую информацию о каждом файле: имя файла, количество блоков, занимаемых файлом на магнитном диске, дата и время создания файла.

Примеры.

1) PIP > /LI < CR >

Распечатка на терминале текущего каталога пользователя.

2) PIP > *.FTN/LI < CR >

Распечатка на терминале всех файлов из текущего каталога пользователя с типом .FTN на устройстве по умолчанию (SY0:).

Очистка каталога

Формат командной строки:

ВВФ1,ВВФ2, ...,ВВФN/PU

Удаляются все версии файлов, кроме последней.

Примеры.

1) PIP > DK1:*.FTN/PU < CR >

Удаление всех версий, кроме последних, файлов типа .FTN из текущего каталога пользователя на устройстве DK1:.

2) PIP > *.*/PU < CR >

Удаление всех версий, кроме последних, для всех файлов из текущего каталога пользователя на устройстве по умолчанию (SY0:).

Определение свободного пространства на диске

Формат командной строки:

УСТР:/FR < CR >

Распечатка свободного пространства (в блоках) на магнитном диске (выводится на терминал).

Примеры.

1) PIP > DK1:/FR < CR >

Распечатать свободное пространство на DK1:.

2) PIP > /FR < CR >

Распечатать свободное пространство на системном устройстве (SY0:).

4.3.12. Диагностика, исправление ошибок. Во время работы пользователя в ОС РВ возможны различные ошибки, которые система обнаруживает и выводит соответствующую диагностику на терминал.

Ошибки, связанные с неверным употреблением команд для интерпретатора MCR или других программ системы (FOR, TKB, PIP), сопровождаются обычно информацией

ILLEGAL COMMAND — неверная команда

Чтобы получить сведения о правильном синтаксисе и семантике команды, применяется команда HELP — единственная, которую пользователь может выполнить до входа в систему. Формат команды

HELP параметр 1, ..., параметр 9

где список параметров определяет список команд, о которых пользователь желает получить информацию. Например, команда

HELP HEL, ABO, BYE

требует сообщить информацию о командах:

HEL — вход в систему,
ABO — снятие задачи с выполнения,
BYE — выход из системы.

Выяснив правильные синтаксис и семантику команды, пользователь должен повторить ее на терминале.

Сообщения об ошибках транслятор с фортрана делит на два класса. Сообщения первой фазы трансляции выводятся в листинге сразу после ошибочного оператора и имеют вид

*****T

где T — буквенный индекс ошибки (табл. 4.2). Ошибки первой фазы трансляции обнаруживаются просмотром части программы.

Таблица 4.2

T	Ошибка
B	Позиции 1—5 строки продолжения не пусты
C	Недопустимое продолжение
E	Отсутствует оператор END
H	Слишком длинная текстовая константа
I	Недопустимый символ
K	Недопустимое определение метки
M	Многократное употребление метки
P	Несоответствие скобок
S	Синтаксическая ошибка
U	Недопустимый оператор

После просмотра всей программы обнаруживаются ошибки второй фазы. Сообщения об ошибках второй фазы выводятся в листинге после текста программы перед картой памяти и имеют вид

IN LINE *n* MSG # *m* текст

где *n* — номер ошибочного оператора, *m* — номер ошибки, текст — краткое описание ошибки.

Если во время трансляции, например, главной программы и подпрограммы PROG 1 были обнаружены ошибки и имеются предупреждения, то на терминал будет выведено сообщение, например, такого вида

```
.MAIN.   ERROS:3   WARNINGS:2
PROG 1   ERROS:1   WARNINGS:1
```

или

```
в главной программе ошибок : 3, предупреждений : 2
в PROG 1            ошибок : 1, предупреждений : 1
```

Предположим, что команда для транслятора имела следующую форму:

```
> FOR PROG,PROG=PROG
```

Тогда после получения на терминале сообщения о наличии ошибок трансляции пользователь должен вывести листинг, например, на терминал

```
>PIP TI:=PROG.LST
```

Просматривая такой текст листинга, пользователь получает информацию о допущенных ошибках, переходит в режим редактирования

```
>EDI PROG.FTN
```

и исправляет их.

Диагностика построителя задач, указывающая на ненормальную форму завершения компоновки файла задачи, запрещает дальнейшее выполнение программы. Для простых по структуре задач, как правило, это связано с отсутствием достаточного непрерывного пространства на диске для образа задачи. В случае такой ситуации необходимо проверить свободное пространство командой

```
>PIP /FR
```

и, если необходимо, осуществить очистку диска командами PIP удаления файлов.

После запуска задачи на выполнения командой

```
>RUN PROG
```

где PROG — имя файла, содержащего образ задачи, возможно появление ошибок вычислений.

Если выполнение программы продолжается сверх ожидаемого интервала времени без диагностики ОС РВ, то, возможно, произошло зацикливание, т. е. в программе имеется ошибочное условие выхода из цикла, которое с имеющимися данными не может быть истиной. В такой ситуации следует прервать выполнение программы командой

```
>ABO PROG
```

где PROG — имя прерываемой задачи. Затем следует проверить фрагменты программы, связанные с организацией циклов.

Диагностика ошибок во время выполнения программы имеет следующий формат:

PROG — EXITING DUE TO ERROR *n*

текст

AT PC = адрес
FCS: файл
IN PROG1 AT *m*1
FROM PROG2 AT *m*2
.....
FROM PROGK AT *mk*

где PROG — имя выполняемой задачи; *n* — номер ошибки, которую объясняет краткий текст;

EXITING DUE TO ERROR — выход по ошибке

AT PC = адрес в счетчике команд во время прерывания арифметических операций с плавающей точкой (деление на нуль, переполнение и т. д.) FCS: файл — имя файла, с которым связана ошибка ввода-вывода, если ошибка связана с вводом — выводом.

IN PROG1 AT *m*1

— ошибка произошла в программной единице с именем PROG1 в операторе номер *m*1, далее указывается цепочка вызовов программных единиц, приведшая к ошибке

из PROG2 вызов в операторе *m*2
.....
из PROGK вызов в операторе *mk*

По этой цепочке вызовов можно пытаться найти причину появления ошибки, анализируя текст программы и требуемую логику ее работы.

Глава 5

МЕТОДЫ ВЫЧИСЛЕНИЙ

● 5.1. Примеры

5.1.1. История и содержание предмета. *Вычислительной математикой* называют раздел математики, в котором изучаются разнообразные проблемы получения числовых результатов решения математических задач.

Если обратиться к истории математики, то можно заметить, что вычислительная математика превратилась в самостоятельную ветвь сравнительно недавно, где-то в середине нашего столетия. Этот факт в любом направлении науки связывается с появлением собственных внутренних задач, которые могут стимулировать исследователей этого направления практически без взаимодействия с соседними.

Вычислительная математика как часть математики имеет столь же древнюю и богатую историю, как и сама математика. Можно утверждать, что почти все результаты математики, которые носили конструктивный, формульный характер, ложились «в копилку» вычислительной математики. Евклидова геометрия и механика Ньютона, теория электромагнитного поля и квантовая теория построена на математической основе и дают мощные инструменты вычислений. Есть математические результаты, которые не могут быть прямо использованы в вычислениях; например, теорема о неразрешимости в радикалах общего алгебраического уравнения выше четвертой степени, теорема существования решения уравнения без предъявления алгоритма его построения и т. п. Следует все-таки признать, что деление математики на «чистую», прикладную, вычислительную отвечает скорее узкой специализации математиков, а не задачам, которые математика призвана решать. Для решения проблем естествознания, техники математика должна рассматриваться единой и неделимой. Только в этом случае использование результатов математики и ее неотъемлемой части — вычислительной математики — будет плодотворным и эффективным.

С появлением ЭВМ начался золотой век вычислительной математики, она бурно развивается. Ее приложения в науке и технике расширяются с каждым годом. Однако появилась реальная опасность отождествления с вычислительной математикой численных методов математики, как наиболее приспособленных к работе на машинах первых четырех поколений.

И до появления ЭВМ инженеры и ученые проводили довольно сложные вычисления и весьма успешно: была построена Эйфелева башня, были рассчитаны положения планет Плутон, Нептун, велись аэродинамические расчеты в самолетостроении. В чем же причина успехов вычислительной математики в те времена? Ответ здесь однозначный: причина успеха в применении всего арсенала математики того времени для решения задачи, в гибком сочетании аналитических и вычислительных методов. Прежде чем считать, необходимо было выполнить большую аналитическую работу — инструменты для вычислений не могли компенсировать, как сейчас, слабые методы вычислений.

Обратившись к периоду развития вычислительной математики после появления ЭВМ, можно увидеть, что наиболее яркие достижения в решении задач были получены именно теми учеными и инженерами, кто работал на ЭВМ, применяя все имеющиеся средства математики: «чистой», прикладной, вычислительной.

С точки зрения техники вычислений математика дает в ее распоряжение методы, которые условно можно разбить на следующие четыре группы: *качественные*, *аналитические*, *методы возмущений* и *численные методы*.

5.1.2. Примеры качественных методов. Примером качественных методов может служить следующая теорема алгебры:

всякий многочлен степени $n \geqslant 1$ с любыми числовыми коэффициентами имеет n корней, считая корень столько раз, какова его кратность.

Эта теорема не дает подходов для нахождения корней, но позволяет вычислителю определить число корней алгебраических уравнений.

Для отыскания замкнутых траекторий (предельных циклов) на плоскости применяется следующая теорема Бендиксона:

пусть в кольцевой области G нет точек x_1, x_2 таких, что $f_1(x_1, x_2)=0$, $f_2(x_1, x_2)=0$,— точек покоя системы

$$\frac{dx_1}{dt}=f_1(x_1, x_2); \quad \frac{dx_2}{dt}=f_2(x_1, x_2).$$

Пусть все траектории этой системы, начинающиеся при $t=0$ на границе G, остаются внутри G при всех $t>0$. Тогда в G имеется по крайней мере один цикл.

Эта теорема относится к качественным методам исследования дифференциальных уравнений.

Теорема линейной алгебры о приводимости матрицы оператора A, действующего в n-мерном пространстве, имеющего n линейно независимых собственных векторов с собственными числами λ_1, λ_2, ..., λ_n, к диагональному виду

$$\begin{pmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ 0 & & \ddots \lambda_n \end{pmatrix}$$

относится также к качественным методам линейной алгебры. Эта теорема указывает канонический (простейший) вид матрицы

оператора в базисе из собственных векторов, но не дает метода построения этого базиса.

5.1.3. Примеры аналитических методов. Примером аналитических методов являются формулы решения квадратного уравнения $x^2+px+qx=0$:

$$x_{1,2}=-p/2\pm\sqrt{(p/2)^2-q}$$

— или формулы Кардано решения кубического уравнения.

Примером аналитических методов решения дифференциальных уравнений является метод разделения переменных. Например, решая этим методом задачу

$$\frac{dy}{dx}=xy^2,\ y(0)=1,\ 0\leqslant x\leqslant 1,$$

имеем формулу

$$y=2/(2-x^2).$$

Решение общих линейных систем обыкновенных дифференциальных уравнений в виде конечного числа формул может быть получено для систем не выше четвертого порядка ($n\leqslant 4$)

$$\frac{dy_i}{dx}=\sum_{j=1}^{n}a_{ij}y_j+f_i(x),\ 1\leqslant i\leqslant n,\ 0\leqslant x\leqslant 1,\ y_i(0)=0$$

и только для тех правых частей $f_i(x)$, которые интегрируются аналитически с множителями $x^m\exp(\lambda x)$. Это легко понять, если учесть, что для получения формул решения необходимо найти собственные числа матрицы $A=a_{ij}$, т. е. решить алгебраическое уравнение $\det(A-\lambda E)=0$ n-й степени, а это можно сделать в конечном виде только для $n\leqslant 4$. Не следует при этом забывать, что для частных линейных систем может оказаться возможным представить решение конечным числом формул и для $n>4$.

Однако аналитические методы не ограничиваются рассмотрением алгоритмов с применением конечного числа формул, вводятся бесконечные процессы, предельные переходы, т. е. весь арсенал математического анализа.

Так, например, аналитическим является метод последовательных приближений Перрона решения рассмотренной выше задачи Коши для системы линейных уравнений

$$y_i^{k+1}(x)=\int_0^x\left[\sum_{j=1}^{n}a_{ij}y_j^{(k)}(s)+f_i(s)\right]ds,\ 1\leqslant i\leqslant n,\ k=0,\ 1,\ \ldots,$$

$$y_i^{(0)}(x)\equiv 0,$$

если правые части $f_i(x)$ допускают аналитическое интегрирование бесконечное число раз, т. е. известны формулы для интегралов

$$\int_0^x f_i(s_0)\,ds_0,\ \int_0^x ds_0\int_0^{s_0}f_i(s_1)\,ds_1\ldots\int_0^x ds_0\int_0^{s_0}ds_1\ldots\int_0^{s_{k-1}}f_i(s_k)\,ds_k.$$

Например, это могут быть многочлены, тригонометрические функции и т. п. Тогда решение $y_i(x)$ записывается в виде (если существует предел)

$$y_i(x)=\lim_{k\to\infty} y_i^k(x),\quad 0\leqslant x\leqslant 1.$$

Аналитическим методом решения систем линейных уравнений

$$Ax=b$$

с определителем $\det A\neq 0$ является правило Крамера

$$x_1=\frac{d_1}{d},\quad x_2=\frac{d_2}{d},\quad \ldots,\quad x_n=\frac{d_n}{d},$$

где d — определитель матрицы A;

$$d=\begin{vmatrix} a_{1,1} \ldots a_{1,j} \ldots a_{1,n} \\ a_{2,1} \ldots a_{2,j} \ldots a_{2,n} \\ \ldots\ldots\ldots\ldots\ldots \\ a_{n,1} \ldots a_{n,j} \ldots a_{n,n} \end{vmatrix};$$

определитель d_j получается из d заменой j-го столбца столбцом из свободных членов.

В тех случаях, когда правило Крамера легко применить (n мало), оно дает явное выражение решения системы через коэффициенты — элементы матрицы A и вектора b. При больших значениях n практически оно неприменимо и провести вычисления с его помощью не представляется возможным, так как количество слагаемых в определителях с ростом n растет как $n!$.

Это один из многочисленных примеров, когда аналитический метод не может быть положен в основу вычислений.

5.1.4. Примеры методов возмущений. Следующая группа методов, названная *методами возмущений*, занимает промежуточное положение между методами, дающими точное и приближенное решение задач. Хотя методы возмущений относятся к аналитическим методам, они выделяются в особое направление как по большому и разнообразному математическому аппарату, так и тому месту в методах вычислений, которые они занимают.

Фактически методы возмущений представляют собой промежуточное, связующее звено между аналитическими и численными методами.

Термин «возмущение» обусловлен тем, что рассматривается обычно задача U_ε, зависящая от малого параметра $|\varepsilon|\ll 1$ (ε может быть вектором), которая является возмущением предельной, невозмущенной ($\varepsilon=0$) задачи U_0. Решение задачи U_0 предполагается известным. Методы возмущений дают различные подходы к решению U_ε, при этом используется малость параметра возмущения ε и информация о решении предельной задачи U_0.

Приведем пример одного из методов возмущений, широко использующегося в вычислениях. Будем рассматривать невозмущенное уравнение, иллюстрирующее аналитические методы.

Пусть требуется решить алгебраическое уравнение

$$U_\varepsilon:\ \varepsilon x^5+x^2+px+q=0. \tag{5.1.1}$$

где ε — малый параметр. Пусть невозмущенное уравнение

$$U_0:\ x^2+px+q=0, \tag{5.1.2}$$

не имеет кратных корней $(p^2/4)-q\neq 0$

$$x_1^{(0)}=-p/2+\sqrt{p^2/4-q},\ \ x_2^{(0)}=-p/2-\sqrt{p^2/4-q}.$$

Для двух из пяти корней уравнения (5.1.1) можно найти формулы в виде рядов по ε (рядов возмущений), в пределе при $\varepsilon\to 0$ дающие корни невозмущенного уравнения (5.1.2): $x_1^{(0)}$, $x_2^{(0)}$.

Метод возмущений сводится к представлению двух корней $x_1(\varepsilon)$, $x_2(\varepsilon)$ (5.1.1) в виде рядов

$$x_1(\varepsilon)=x_1^{(0)}+\varepsilon x_1^{(1)}+\varepsilon^2 x_1^{(2)}+\varepsilon^3 x_1^{(3)}+\ldots, \tag{5.1.3}$$

$$x_2(\varepsilon)=x_2^{(0)}+\varepsilon x_2^{(1)}+\varepsilon^2 x_2^{(2)}+\varepsilon^3 x_2^{(3)}+\ldots \tag{5.1.4}$$

с неизвестными коэффициентами $x_1^{(i)}$, $x_2^{(i)}$, $i=1, 2, \ldots$. Подставим (5.1.3), (5.1.4) в (5.1.1), приравняем коэффициенты при одинаковых степенях малого параметра ε и, таким образом, получим выражения для последовательного определения $x_1^{(i)}$, $x_2^{(i)}$. Например, при ε^1 имеем

$$\varepsilon\left(x_1^{(0)}\right)^5+\varepsilon 2x_1^{(0)}x_1^{(1)}+\varepsilon p x_1^{(1)}=0,$$

$$\varepsilon\left(x_2^{(0)}\right)^5+\varepsilon 2x_2^{(0)}x_2^{(1)}+\varepsilon p x_2^{(1)}=0.$$

Отсюда находим

$$x_1^{(1)}=\frac{(-1)}{2x_1^{(0)}+p}\left(x_1^{(0)}\right)^5=-\frac{(-p/2+\sqrt{p^2/4-q})^5}{2\sqrt{p^2/4-q}},$$

$$x_2^{(1)}=\frac{(-1)}{2x_2^{(0)}+p}\left(x_2^{(0)}\right)^5=-\frac{(-p/2-\sqrt{p^2/4-q})^5}{2\sqrt{p^2/4-q}}.$$

Эти формулы дают выражения для двух корней (5.1.1) в первом приближении

$$x_1(\varepsilon)=-p/2+\sqrt{p^2/4-q}-\varepsilon\frac{(-p/2+\sqrt{p^2/4-q})^5}{2\sqrt{p^2/4-q}}+O(\varepsilon^2), \tag{5.1.5}$$

$$x_2(\varepsilon)=-p/2+\sqrt{p^2/4-q}-\varepsilon\frac{(-p/2-\sqrt{p^2/4-q})^5}{2\sqrt{p^2/4-q}}+O(\varepsilon^2). \tag{5.1.6}$$

Можно доказать сходимость рядов (5.1.3), (5.1.4) для достаточно малых значений $|\varepsilon|$. Однако формулы для приближений выше первого становятся громоздкими и на практике редко применяются.

Ряды (5.1.3), (5.1.4) определяют аналитическую функцию по ε, а методы возмущений, приводящие к таким рядам, называются *регулярными методами возмущений.*

Заметим, что оставшиеся три корня (5.1.1) не могут быть определены функциями без особенностей по ε. Три оставшихся

корня $x_3(\varepsilon)$, $x_4(\varepsilon)$, $x_5(\varepsilon)$ по ε должны иметь особенность при $\varepsilon=0$, т. е.

$$\lim_{\varepsilon\to 0} x_j(\varepsilon)=\infty,\quad j=3,\ 4,\ 5.$$

Чтобы найти характер этой особенности, произведем в (5.1.1) замену

$$x=\frac{1}{\varepsilon^{\nu}}y.$$

Получим

$$\frac{1}{\varepsilon^{5\nu-1}}y^5+\frac{1}{\varepsilon^{2\nu}}y^2+\frac{p}{\varepsilon^{\nu}}y+q=0. \tag{5.1.7}$$

Найдем показатель особенности ν из условия, чтобы коэффициенты при старших степенях y в (5.1.7) имели одну и ту же особенность, т. е.

$$5\nu-1=2\nu,\quad \nu=1/3.$$

Тогда замена приведет уравнение (5.1.1) к уравнению

$$V_\varepsilon:\ y^5+y^2+\varepsilon^{1/3}py+\varepsilon^{2/3}q=0,$$

для которого невозмущенное уравнение имеет вид

$$V_0:\ y^5+y^2=0.$$

Выбирая из решений V_0 ненулевые, находим

$$y_3^{(0)}=-1,\quad y_4^{(0)}=\frac{1+\sqrt{3}i}{2},\quad y_5^{(0)}=\frac{1-\sqrt{3}i}{2}.$$

Теперь к задаче V_ε можно применить регулярный метод возмущений. Ищем три корня $V_\varepsilon-y_3(\varepsilon)$, $y_4(\varepsilon)$, $y_5(\varepsilon)$ в виде рядов по степеням $\varepsilon^{1/3}$:

$$y_3(\varepsilon)=y_3^{(0)}+\varepsilon^{1/3}y_3^{(1)}+\varepsilon^{2/3}y_3^{(2)}+\varepsilon^{3/3}y_3^{(3)}+\ldots,$$
$$y_4(\varepsilon)=y_4^{(0)}+\varepsilon^{1/3}y_4^{(1)}+\varepsilon^{2/3}y_4^{(2)}+\varepsilon^{3/3}y_4^{(3)}+\ldots,$$
$$y_5(\varepsilon)=y_5^{(0)}+\varepsilon^{1/3}y_5^{(1)}+\varepsilon^{2/3}y_5^{(2)}+\varepsilon^{3/3}y_5^{(3)}+\ldots.$$

Так же как и выше, подставим эти ряды в V_ε, приравняем коэффициенты при одинаковых степенях $\varepsilon^{1/3}$ и определим $y_3^{(i)}$, $y_4^{(i)}$, $y_5^{(i)}$, $i=1,\ 2,\ \ldots$. При $\varepsilon^{1/3}$ имеем:

$$\varepsilon^{1/3}\cdot 5\left(y_3^{(0)}\right)^4y_3^{(1)}+\varepsilon^{1/3}2y_3^{(0)}y_3^{(1)}+\varepsilon py_3^{(0)}=0,$$
$$\varepsilon^{1/3}\cdot 5\left(y_4^{(0)}\right)^4y_4^{(1)}+\varepsilon^{1/3}2y_4^{(0)}y_4^{(1)}+\varepsilon^{1/3}py_4^{(0)}=0,$$
$$\varepsilon^{1/3}\cdot 5\left(y_5^{(0)}\right)^4y_5^{(1)}+\varepsilon^{1/3}2y_5^{(0)}y_5^{(1)}+\varepsilon^{1/3}py_5^{(0)}=0.$$

Отсюда находим

$$y_3^{(1)}=\frac{p}{3},\quad y_4^{(1)}=-\frac{p}{5\left(y_4^{(0)}\right)^3+2},\quad y_5^{(1)}=-\frac{p}{5\left(y_5^{(0)}\right)^3+2}.$$

Учитывая связь между x и y, запишем приближения для трех корней уравнения (5.1.1) с точностью до $O(\varepsilon^{1/3})$:

$$x_3(\varepsilon)=\frac{1}{\varepsilon^{1/3}}\left(-1+\varepsilon^{1/3}\frac{p}{3}\right)+O(\varepsilon^{1/3}), \tag{5.1.8}$$

$$x_4(\varepsilon)=\frac{1}{\varepsilon^{1/3}}\left(\frac{1+\sqrt{3}i}{2}-\varepsilon^{1/3}\frac{p}{5\left(1+\frac{\sqrt{3}i}{2}\right)^3+2}\right)+O(\varepsilon^{1/3}), \tag{5.1.9}$$

$$x_5(\varepsilon)=\frac{1}{\varepsilon^{1/3}}\left(\frac{1+\sqrt{3}i}{2}-\varepsilon^{1/3}\frac{p}{5\left(\frac{1-\sqrt{3}i}{2}\right)^3+2}\right)+O(\varepsilon^{1/3}). \tag{5.1.10}$$

Ряды вида (5.1.8)—(5.1.10) определяют функции $x_3(\varepsilon)$, $x_4(\varepsilon)$, $x_5(\varepsilon)$ с особенностью по ε. Методы возмущений, приводящие к таким рядам, называются *сингулярными методами возмущений.*

Прием, который сводит сингулярный метод возмущений к регулярному, называют *регуляризацией.* Замена $x=\varepsilon^{-\nu}y$ есть регуляризация.

Можно доказать сходимость рядов в (5.1.8)—(5.1.10) для достаточно малых $|\varepsilon|$.

Итак, методы возмущений позволили для алгебраического уравнения 5-й степени (5.1.1) записать при малых значениях $|\varepsilon|$ приближенные формулы всех пяти корней (5.1.5), (5.1.6), (5.1.8)—(5.1.10).

Таким образом, методы возмущений расширяют возможности аналитических методов и включают в сферу приложений те классы задач, которые могут быть приближенно решены аналитически.

Наконец, когда в уравнении (5.1.1) значение $|\varepsilon|$ не мало, методы возмущений неприменимы. Точное значение ε_0, при котором метод возмущений перестает работать, связано с оценкой погрешности формул для корней $x_i(\varepsilon)$, $1\leqslant i\leqslant 5$ и с требуемой точностью вычисления корней.

В диапазоне изменения ε, в котором не могут применяться методы возмущений, используются численные методы.

5.1.5. Пример численного метода. *Численные методы* — это такие методы решения задач, которые сводятся или могут быть сведены к арифметическим действиям над числами.

Рассмотрим задачу

$$\frac{dy}{dx}=xy^2+\varepsilon\sin x,\quad y(0)=1,\quad 0\leqslant x\leqslant 1, \tag{5.1.11}$$

для которой невозмущенная ($\varepsilon=0$) задача имеет аналитическое решение, приведенное выше. Для малых ε можно найти ряд возмущений по аналогии с задачей (5.1.1):

$$y(x,\varepsilon)=y^{(0)}(x)+\varepsilon y^{(1)}(x)+O(\varepsilon^2), \tag{5.1.12}$$

где $y^{(0)}(x)=\frac{2}{2-x^2}$, $y^{(1)}(x)$ определяется решением задачи

$$\frac{dy^{(1)}}{dx}=2xy^{(0)}(x)y^{(1)}+\sin x,\quad y^{(1)}(0)=0,$$

откуда

$$y^{(1)}(x)=\int_0^x e^{\int_s^x \frac{4s_1}{2-s_1}ds_1}\sin s\,ds.$$

Последний интеграл можно вычислить аналитически, используя элементарные функции. Но если возмущающая функция $\varepsilon u(x, y)$ такова, что интеграл для $y^{(1)}(x)$ не выражается через известные функции, то для вычислений $y(x, \varepsilon)$ по формулам (5.1.12) в какой-либо точке $x_0\in[0, 1]$ необходимо применять численные методы. Это пример сочетания аналитического метода и метода возмущений.

При $\varepsilon=1$ к задаче (5.1.11) методы возмущений уже не применяются. Задача (5.1.11) является типичным объектом численного анализа. К ней можно применить, например, известный из курса высшей математики явный метод Эйлера. Приближенные значения y_i к точному решению в точках $x_i=ih$, $i=0, 1, 2, \ldots, N$, $h={}^1/_N$ находятся из соотношений

$$y_{i+1}=y_i+h(x_iy_i^2+\sin x_i),\quad 0\leqslant i\leqslant N,\quad y_0=1. \qquad (5.1.13)$$

Погрешность приближений имеет оценку

$$\max_{0\leqslant i\leqslant N}|y(x_i)-y_i|=O(h),\quad h\to 0.$$

Обратим внимание на то, что метод Эйлера (5.1.13) сводит задачу приближенного определения точного решения $y(x_i)$ в точках x_i к арифметическим операциям и вычислению элементарной функции $\sin x_i$, т. е. это действительно численный метод, так как вычисление $\sin x_i$ также можно свести к арифметическим операциям над числами.

5.1.6. Общие замечания. Подводя итог введению, следует заметить, что, как правило, на практике приходится иметь дело с задачами, зависящими от одного или нескольких параметров, начальных данных и т. п., которые лежат в некоторых интервалах своего изменения.

Для одних параметров, возможно, удается применить регулярный метод возмущений, для других — сингулярный, третьих — численный метод. Наилучший эффект достигается при сочетании всех рассмотренных подходов. Во-первых, результаты применения должны совпадать с учетом точности вычислений в общей области действия методов; это хороший контроль правильности проведенных вычислений. Во-вторых, к любой задаче следует подходить по принципу «сверху вниз», так же как в алгоритмизации и программировании. Сначала математическая задача изучается

качественными методами, затем аналитическими, далее методами возмущений и, наконец, численными методами.

Численным методам в данной книге уделено основное внимание. Как отмечалось выше, успех численных методов объясняется их сравнительно простой реализацией на ЭВМ.

Однако представляется целесообразным перед тем, как применять численные методы к задаче, ответить на следующие вопросы:

1. Какая «ближайшая» задача решается аналитически?
2. Нельзя ли рассматриваемую задачу считать возмущенной «близкой», решаемой аналитически?
3. Какая «ближайшая» задача решается успешно численно и каким методом?

Частичным ответом на первый вопрос является само определение «близкой» задачи; что под этим термином понимается в конкретной математической модели подробнее см. в п. 5.2. Следующей частью является фактическое построение аналитического решения.

Затем, если параметры модели и требуемая точность вычислений позволяют применить метод возмущений (положительный ответ на второй вопрос), то его применяют, приняв аналитическое решение за нулевое приближение для решения исходной задачи. Если метод возмущений нельзя применять, то «близкая» задача (аналитическое решение) может служить тестовой задачей для контроля разрабатываемого численного метода.

Если «близкая» задача решается только численно и известен метод ее решения (получен ответ на третий вопрос), то следует пытаться решать исходную задачу, сочетая численный метод и метод возмущений. Если эту комбинацию не удается применить, то «близкая» задача (численное решение) может служить тестом для контроля разрабатываемого численного метода.

Все эти «меры предосторожности» кажутся излишними, тем более что численные методы просто реализовать. Но эта простота часто имеет обманчивый вид, так как возникает очень трудная практическая оценка погрешности вычислений.

Поэтому *искусство вычислений состоит фактически не столько в предъявлении числовых результатов в виде таблиц чисел, графиков, сколько в обосновании того, что эти результаты получены с заданной точностью.*

● 5.2. Масштабирование и замена переменных

5.2.1. Анализ размерностей. Безразмерные переменные. Выше использовались термины «близкая» задача, параметр возмущения. Покажем, как определяется параметр или параметры возмущения, выясним, какая задача является «близкой» к данной задаче, что следует сделать с конкретной прикладной задачей еще до этапа ее решения каким-либо методом.

Конкретная техническая задача при своей математической формулировке записывается в виде соотношений, которые содержат переменные, константы в обозначениях и размерностях, принятых в той области техники, к которой эта задача относится.

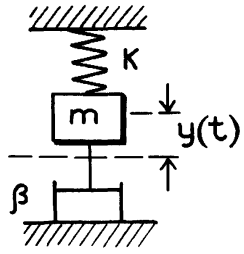

Рис. 5.1

Пример 1. В механике свободное движение массы m, закрепленной на пружине с коэффициентом жесткости k и демпфером с коэффициентом вязкости β (рис. 5.1), описывается уравнением

$$m\frac{d^2y}{dt^2}+\beta\frac{dy}{dt}+ky=0. \tag{5.2.1}$$

Пусть в момент времени $t=0$ заданы начальное смещение $y(0)$, начальная скорость $\frac{dy}{dt}(0)=0$. Тогда найти движение на интервале времени $0\leqslant t\leqslant T$ означает проинтегрировать уравнение (5.2.1) с начальными условиями

$$y(0)=y_0,\ \frac{dy}{dt}(0)=0. \tag{5.2.2}$$

В конкретной задаче масса m может выражаться в граммах (г), y — в сантиметрах (см), t — в секундах (с), β — в г/с, k — в г/с^2. Например, $y_0=5$ см, $m=50$ г, $T=20$ с, $\beta=10$ г/с, $k=200$ г/с^2. В размерных переменных t, y трудно обнаружить, какая задача близка к рассматриваемой — та, в которой можно пренебречь демпфированием, или та, где можно пренебречь собственными колебаниями.

Перейдем к безразмерным переменным τ, x с помощью замены

$$\tau=\sqrt{k/m}t,\ \ x=y/y_0. \tag{5.2.3}$$

Для этого введем величину $\omega_0=\sqrt{k/m}$, равную собственной частоте колебаний без демпфирования. Смещение x теперь выражается в единицах начального смещения. Замена по формулам (5.2.3) дает

$$\frac{dy}{dt}=\frac{d(y_0x)}{d\tau}\frac{d\tau}{dt}=\frac{dx}{d\tau}y_0\omega_0;\ \ \frac{d^2y}{dt^2}=\frac{d^2x}{d\tau}y_0^2\omega_0^2$$

и приводит к уравнению в безразмерных переменных x, τ

$$\frac{d^2x}{d\tau^2}+\mu\frac{dx}{d\tau}+x=0,\ \ 0\leqslant\tau\leqslant\omega_0T, \tag{5.2.4}$$

где $\mu=\beta/(m\omega_0)$, с начальными условиями

$$x(0)=1,\ \frac{dx}{d\tau}=0. \tag{5.2.5}$$

Для рассматриваемых значений параметров имеем $\mu=0{,}1$. Поэтому можно попытаться на интервале $0\leqslant t\leqslant 1$ приближенное решение (5.2.4) находить методом возмущений, считая невозмущенной задачу (5.2.4), (5.2.5) с $\mu=0$ (т. е. близкая задача — это задача без демпфирования). Заметим, что этот факт обнаруживается не по значению коэффициента вязкости β, а по комбинации трех параметров $\mu=\beta/(m\omega_0)$. Кроме того, важна длина интервала изменения τ; при больших значениях $\omega_0 T\geqslant 1$ задачу без демпфирования также нельзя считать близкой. Подробно этот факт рассматривается ниже.

Фактически безразмерный параметр μ является параметром подобия. Системы (5.2.1) с разными значениями β, m, ω_0, но с одинаковыми μ ведут себя одинаково и могут быть исследованы интегрированием одного и того же уравнения (5.2.4).

Пример 1 носит иллюстративный характер, поскольку решение задачи (5.2.1), (5.2.2) выражается в элементарных функциях, но следующее небольшое изменение задачи и приведение ее к безразмерному виду уже имеет содержательный смысл.

Пример 2. Пусть сила упругости пружины нелинейно зависит от смещения

$$F=-ky-k_3y^3,$$

где k, k_3 — постоянные. При этом движение массы m описывается уравнением

$$m\frac{d^2y}{dt^2}+\beta\frac{dy}{dt}+ky+k_3y^3=0 \qquad (5.2.6)$$

с условиями (5.2.2).

Если ввести те же безразмерные переменные, что и в примере 1 с помощью замены (5.2.3), то получим уравнение

$$\frac{d^2x}{d\tau^2}+\mu\frac{dx}{d\tau}+x+\varepsilon x^3=0, \qquad (5.2.7)$$

где $\mu=\dfrac{\beta}{m\omega_0}$, $\varepsilon=\dfrac{k_3y_0^2}{k}$, с начальными условиями (5.2.5).

В уравнении (5.2.7) имеем еще один безразмерный параметр ε, который характеризует влияние нелинейности. Если параметр ε мал, то при определенных условиях можно применять метод возмущений по ε.

Как уже замечено выше, в задачах (5.2.4), (5.2.7) присутствует важный безразмерный параметр $\omega_0 T$, характеризующий длину интервала наблюдения или интегрирования, от значения которого зависит возможность применения методов возмущений. От этого параметра можно освободиться масштабированием — переходом к новой независимости переменной

$$t_1=\tau/(\omega_0 T).$$

Тогда интервал наблюдения в переменных x, t_1 будет единичным, $0 \leqslant t_1 \leqslant 1$. Но при этом уравнения (5.2.4), (5.2.7) примут вид

$$\frac{1}{\omega_0 T^2}\frac{d^2x}{dt_1^2}+\frac{\mu}{\omega_0 T}\frac{dx}{dt_1}+x=0;$$

$$\frac{1}{\omega_0 T^2}\frac{d^2x}{dt_1^2}+\frac{\mu}{\omega_0 T}\frac{dx}{dt_1}+x+\varepsilon x^3=0,$$

или после введения новых параметров $\mu_2=(\omega_0 T)^{-2}$, $\mu_1=\mu(\omega_0 T)^{-1}$ — следующий вид:

$$\mu_2\frac{d^2x}{dt_1^2}+\mu_1\frac{dx}{dt_1}+x=0;\quad \mu_2\frac{d^2x}{dt_1^2}+\mu_1\frac{dx}{dt_1}+x+\varepsilon x^3=0.$$

Теперь вся характеристика задачи, например (5.2.7), содержится в трех безразмерных параметрах ε, μ_1, μ_2, в то время как в исходной формулировке имеется шесть размерных параметров: y_0, m, T, β, k, k_3. Это первое преимущество использования безразмерных переменных и масштабирования. Второе состоит в возможности по параметрам ε, μ_1, μ_2 определить, можно ли применять метод возмущений или нет и какая задача близка к рассматриваемой. Если возможно применение только численных методов, то эти же параметры могут определить, какой из методов следует использовать.

Пример 3. Установившееся обтекание пластины потоком вязкой несжимаемой жидкости описывается уравнениями

$$\begin{cases} \dfrac{\partial u}{\partial x}+\dfrac{\partial v}{\partial y}=0, \\ \rho\left(u\dfrac{\partial u}{\partial x}+v\dfrac{\partial u}{\partial y}\right)=-\dfrac{\partial p}{\partial x}+\mu\left(\dfrac{\partial^2 u}{\partial x^2}+\dfrac{\partial^2 u}{\partial y^2}\right), \\ \rho\left(u\dfrac{\partial v}{\partial x}+v\dfrac{\partial v}{\partial y}\right)=-\dfrac{\partial p}{\partial y}+\mu\left(\dfrac{\partial^2 v}{\partial x^2}+\dfrac{\partial^2 v}{\partial y^2}\right) \end{cases} \qquad (5.2.8)$$

с граничными условиями (рис. 5.2)

$$\begin{cases} u(x,0)=0,\ v(x,0)=0,\ x\geqslant 0, \\ \lim\limits_{x\to-\infty} u(x,y)=v_\infty,\ \lim\limits_{x\to-\infty} v(x,y)=0, \end{cases} \qquad (5.2.9)$$

где $u(x, y)$, $v(x, y)$ — проекции вектора скорости жидкости в точке (x, y) на оси x и y соответственно, $p(x, y)$ — давление, ρ — постоянная плотность, μ — коэффициент вязкости жидкости.

Приведение задачи (5.2.8), (5.2.9) к безразмерному виду осуществляется выбором характерного размера L — расстояния

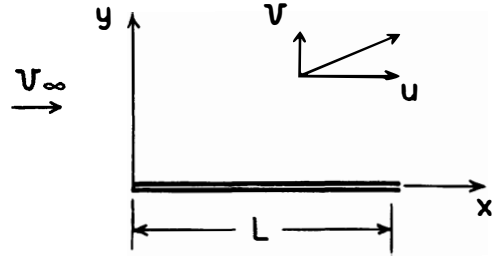

Рис. 5.2

от передней кромки пластины до точки на пластине, где изучается течение: $0 \leqslant x \leqslant L$.

Безразмерными переменными являются независимые u_1, v_1, p_1, зависимые x_1, y_1 и вводятся с помощью формул

$$u_1 = \frac{u}{v_\infty}, \quad v_1 = \frac{v}{v_\infty}, \quad p_1 = \frac{p}{\rho v_\infty^2}, \quad x_1 = \frac{x}{L}, \quad y_1 = \frac{y}{L}.$$

Соотношения (5.2.8), (5.2.9) в безразмерных переменных принимают вид

$$\frac{\partial u_1}{\partial x_1} + \frac{\partial v_1}{\partial y_1} = 0,$$

$$u_1 \frac{\partial u_1}{\partial x_1} + v_1 \frac{\partial u_1}{dy_1} = -\frac{\partial p_1}{dx_1} + \frac{1}{\mathrm{Re}}\left(\frac{\partial^2 u_1}{\partial x_1^2} + \frac{\partial^2 u_1}{\partial y_1^2}\right),$$

$$u_1 \frac{\partial u_1}{\partial x_1} + v_1 \frac{\partial v_1}{dy_1} = -\frac{\partial p_1}{dx_1} + \frac{1}{\mathrm{Re}}\left(\frac{\partial^2 v_1}{\partial x_1^2} + \frac{\partial^2 v_1}{\partial y_1^2}\right),$$

$$u_1(x_1, 0) = 0, \quad v_1(x_1, 0) = 0,$$

$$\lim_{x_1 \to -\infty} u_1(x_1, y_1) = 1, \quad \lim_{x_1 \to -\infty} v_1(x_1, y_1) = 0,$$

где присутствует лишь один безразмерный параметр

$$\mathrm{Re} = \rho \frac{v_\infty L}{\mu},$$

называемый *числом Рейнольдса*. Теперь в зависимости от значений Re в конкретной задаче близкими уравнениями могут быть: при больших значениях Re — уравнения пограничного слоя (параметр возмущения $(\mathrm{Re})^{-1}$), при малых значениях Re — течение Стокса — Озеена (параметр возмущения Re).

Если в конкретной задаче Re не мал и не велик, то применяется численный метод, но контроль результатов осуществляется сравнением вычислений при малых и больших Re с известными течениями. В случае их совпадения в пределах точности вычислений можно надеяться на правильность результатов и в промежуточном диапазоне значений Re. В заключение дадим практическую рекомендацию: *исходная задача должна быть преобразована к безразмерному виду, выделены безразмерные параметры, изучены решения для предельных значений этих параметров.* Часто предельные решения можно получить аналитическими методами.

5.2.2. Замена переменных. Рассмотренные выше преобразования уравнений к безразмерному виду и масштабирование — простейшие линейные замены переменных, которые выполняются в первую очередь. Затем для упрощения вычислений следует пытаться выполнить более сложные нелинейные замены. Наибольшие вычислительные трудности вызывает наличие у искомых решений $y(x)$ областей с большими значениями модуля производных $\left|\frac{dy}{dx}\right|$ или

$\|\operatorname{grad} y(x)\|$ для функций многих переменных $y(x_1, x_2, ..., x_n)$ (рис. 5.3). Можно определенно утверждать, что переменные y, x не являются наилучшими для представления решения. Более подходят для таких функций переменные y, s, где s — длина дуги кривой $y(x)$, т. е.

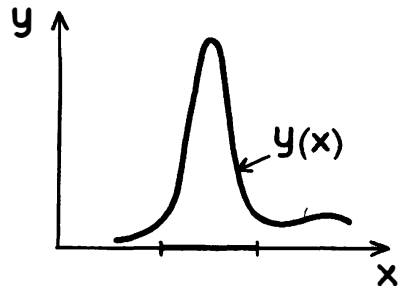

Рис. 5.3

$$ds^2=(dx)^2+(dy)^2,\quad \frac{ds}{dx}=\sqrt{1+(y'_x)^2}.$$

Например, задача

$$\frac{dy}{dx}=1+y^2,\quad y(0)=0,\quad 0\leqslant x<\frac{\pi}{2}$$

имеет решение $y(x)=\operatorname{tg} x$, у которого в точке $x=\pi/2$ особенность. В то же время это уравнение в переменных y, s, имеет вид

$$\frac{dy}{ds}\frac{ds}{dx}=1+y^2,\quad \frac{dx}{ds}=\frac{y'_s}{1+y^2}$$

или

$$\frac{dy}{ds}=\frac{1+y^2}{\sqrt{1+y'^2_x}}=\frac{1+y^2}{\sqrt{1+(1+y^2)^2}}$$

и та же задача с условиями $y(0)=0$, $0\leqslant s<\infty$ имеет решение $y=y(s)$ без особенностей по s.

В более общей форме рекомендуется переход к параметрическому представлению решений

$$y=\varphi(s, t),\quad x=\psi(s, t),$$

который при надлежащем выборе φ, ψ может упростить вычисления в задаче.

Во многих технических задачах приходится решать уравнение Лапласа

$$\Delta\varphi=0,\quad \Delta=\frac{\partial}{\partial x^2}+\frac{\partial}{\partial y^2}$$

в некоторой области D с условиями на границе D. Имеется много методов решения этих задач, но не следует забывать и старый метод конформных преобразований (фактически замен переменных) области D на другую, например единичный круг, для которой известно аналитическое решение. Тогда решение исходной задачи сводится к поиску замены переменных

$$x=x(s, t),\quad y=y(s, t).$$

Методом конформных преобразований великолепно владел советский математик М. В. Келдыш (1911—1978), который

в 40-е годы решил сложные задачи аэродинамики, используя этот подход.

Итак, *математическая формулировка технической задачи не должна рассматриваться как объект, не подлежащий изменениям. Наоборот, задачу следует с помощью эквивалентных преобразований привести к виду, наиболее удобному для ее решения.*

●5.3. Аналитические методы

5.3.1. Определение. Выше отмечалась важность аналитических методов в практике вычислений. Мы будем понимать под аналитическими методы решения задач, представляющие решения через элементарные или специальные функции, для которых могут быть вычислены значения требуемой точности по имеющимся на ЭВМ программам или по таблицам, с применением бесконечных процессов, предельного перехода и арифметических операций над числами. На практике, естественно, бесконечные процессы аналитических решений заменяются конечными с оценкой вносимой при этом погрешности вычисления точного решения.

Приведем простой пример, когда известное аналитическое решение не может быть использовано в вычислениях. Пусть необходимо решить задачу

$$\frac{d^2y}{dx^2}+y=0,\ y(0)=0,\ \frac{dy}{dx}(0)=1,\ 0\leqslant x\leqslant 1.$$

Решение может быть записано в виде

$$y=\sin x.$$

Представим, что требуется решить поставленную задачу с точностью до 20 знаков после запятой. Тогда, например, если на ЭВМ вычисления с двойной точностью дают не более 17 знаков после запятой, заключаем, что аналитическое решение $y=\sin x$ — всего лишь символическая запись ответа. Для вычислений с указанной точностью такое решение неприменимо.

5.3.2. Вычисления с элементарными функциями. Если решение задачи выражаются через элементарные функции ($\sqrt{x}$, e^x, $\sin x$, $\ln x$ и т. п.), то при вычислениях необходимо соблюдать два правила.

1) *Значения аргумента функций с ограниченной областью определения должны проверяться на принадлежность этой области.* Например,

если $x>0$, то $y=\ln x$,

иначе — вывод сообщения, что $x\leqslant 0$.

2) *Значения функций в окрестности точек, где требуется вычислять предел, раскрывая неопределенности вида*

$$\frac{0}{0},\ \frac{\infty}{\infty},\ 0\cdot\infty,\ \infty-\infty,\ 0^0,\ \infty^0,\ 1^\infty,$$

должны вычисляться по формулам, где эти неопределенности уже раскрыты. Например, вычислять:

$$y=\frac{\sin x}{x},\quad -1<x<+1,$$

следует в зависимости от требуемой точности по формулам

$$y=\begin{cases}\dfrac{\sin x}{x}, & |x|\geqslant\delta,\\ 1-\dfrac{x^2}{3!}+O(x^4), & |x|<\delta,\end{cases}$$

где δ — малое число, определяемое точностью вычислений.

Необходимо обратить внимание на область значений функций, участвующих в вычислениях. Если значения выходят за пределы допустимых чисел для ЭВМ (слишком большие по модулю), то счет прекращается аварийным остановом и выдачей на терминал соответствующего сообщения (переполнение).

Эта ситуация часто может быть устранена проведением соответствующего масштабирования в задаче. Пусть, например, необходимо вычислить с некоторым шагом по x

$$y(x)=e^x,\quad 0\leqslant x\leqslant 100.$$

Учитывая, что для y значения в ЭВМ не должны превышать 10^{19}, будем вычислять функцию $y(x)$ по формулам

$$y=\begin{cases}e^x, & 0\leqslant x\leqslant 50,\\ ke^{x-50}, & 50\leqslant x\leqslant 100,\end{cases}$$

с масштабирующим множителем $k=e^{50}$ (без умножения на ЭВМ ke^{x-50}).

Кроме того, выход чисел за допустимые пределы устраняется переходом к другим координатам заменой переменных. В координатах z, x, где $z=\ln y$, исходная функция записывается в виде

$$z(x)=x,\quad 0\leqslant x\leqslant 100,$$

и трудности вычисления $z(x)$ на ЭВМ устранены.

Аналогичная проблема возникает, когда числовые значения слишком малы по модулю (меньше $\sim 10^{-19}$). В этом случае счет не прекращается, но такое число заменяется нулем. Поэтому результат вычислений на ЭВМ выражения

$$y=e^{-x}e^{x/2}e^{x/2}$$

в точке $x=100$ есть нуль, а не единица, так как первый сомножитель заменится нулем. Устранение слишком малых чисел из вычислений осуществляется, так же как и слишком больших чисел, масштабированием и заменой переменных.

5.3.3. Вычисление специальных функций. Если аналитическое решение задачи выражается через специальные функции ($Si(x)$, $I_0(x)$ и т. п.) и в библиотеке программ есть программы их вычисления, то обращение к ним не отличается от вычислений элементарных функций.

Если же в библиотеке отсутствует соответствующая программа, то следует рассмотреть два возможных варианта организации вычисления. Первый заключается в том, чтобы ввести в память ЭВМ таблицу этой функции в интересующем нас диапазоне значений аргумента. Затем вычисления осуществляются по таблице с помощью интерполяции. Второй вариант состоит в представлении этой функции через имеющиеся в библиотеке элементарные и специальные с заданной точностью в аналитическом виде, а затем программировании полученных формул. Различные приближения для специальных функций представлены в [22, 23].

5.3.4. Вычисление элементарных функций комплексного переменного. Эти функции вычисляются либо прямым обращением к библиотечным с соответствующим описанием аргумента и значения функций как комплексных переменных, либо (в случае отсутствия их в библиотеке) — программированием вычислений. Так как функция комплексного переменного $w=f(z)$ может быть представлена в виде ($w=u+iv$, $z=x+iy$)

$$u=u(x,y),\quad v=v(x,y),$$

то определение $u(x,y)$, $v(x,y)$ сводится к разделению вещественной и мнимой частей заданной функции; они выражаются через элементарные функции вещественных аргументов, и программирование не вызывает затруднений. Например, функция Жуковского

$$f(z)=\frac{1}{2}\left(z+\frac{1}{z}\right)$$

вычисляется разделением на вещественную и мнимую части

$$u+iv=\frac{1}{2}\left(x+iy+\frac{1}{x+iy}\right)\frac{x-iy}{x-iy},$$

$$u=\frac{1}{2}\left(x+\frac{x}{x^2+y^2}\right),\quad v=\frac{1}{2}\left(y-\frac{y}{x^2+y^2}\right)$$

и объединением двух вещественных величин в комплексную с помощью библиотечной функции

$$\mathrm{W=CMPLX(U,V)}.$$

Выделение действительной части x у комплексного числа z производится функцией X=REAL(Z) мнимой части — функцией Y= =AIMAG(Z).

5.3.5. Аппроксимация функций. Приближение функций, заданной в аналитической форме другой или другими функциями, определяется в основном необходимостью ускорить процесс вычислений функций с заданной точностью.

Например, время вычисления функции

$$y(x)=\frac{\sin x+\cos x}{e^x+\ln(1+x)} \tag{5.3.1}$$

в точках $0\leqslant x\leqslant 0{,}1$ с точностью ε можно значительно уменьшить, если заменить $y(x)$ приближенно полиномом

$$P_n(x)=a_0+a_1x+\ldots+a_nx^n.$$

Коэффициенты и погрешность можно найти, например, из ряда Тейлора для $y(x)$. Допустим, что ε таково, что полином третьей степени обеспечивает точность ε, т. е.

$$\max_{0\leqslant x\leqslant 0{,}1}|y(x)-P_3(x)|<\varepsilon.$$

Тогда для вычисления $y(x)$ в любой точке $x\in[0{,}01]$ с помощью $P_3(x)$ требуется три операции сложения и три умножения. Этот вычислительный процесс займет гораздо меньше времени счета на ЭВМ, чем вычисление по исходной (5.3.1) формуле для $y(x)$.

На интервалах большой длины целесообразно приближать $y(x)$ рациональными функциями

$$w_{n,m}(x)=\frac{a_0+a_1x+\ldots+a_nx^n}{b_0+b_1x+\ldots+b_mx^m}.$$

Можно область изменения x разбивать на интервалы и на разных интервалах аппроксимировать $u(x)$, либо $P_n(x)$, либо $w_{n,m}(x)$.

5.3.6. Интегрирование. Методы аналитического интегрирования подробно изучаются в курсе высшей математики. Зная таблицу основных интегралов и правила интегрирования, путем замены переменных, интегрирования по частям и т. п. можно вычислить точно интеграл

$$\int_a^b y(x)\,dx=F(b)-F(a),$$

зная первообразную $F(x)$ для широких классов функций. Причем если $F(x)$ выражается через элементарные или специальные функции, для которых можно найти $F(a)$, $F(b)$ с заданной точностью, то считается, что аналитическое интегрирование выполнено. Например,

$$\int_0^1 xe^x\,dx=1,\quad \int_0^1 e^{-x^2}\,dx=\frac{\sqrt{2}}{2}\operatorname{erf}(1),\quad \int_0^1 \frac{\sin x}{x}\,dx=Si(1). \tag{5.3.2}$$

В то же время интеграл от простой функции

$$\int_0^1 \frac{x}{\sin x}\,dx$$

не выражается в конечном виде через значения известных элементарных и специальных функций. Тем более это будет правилом для более сложных подынтегральных функций.

Интеграл можно вычислять численно. Но здесь, как и в аппроксимации функций, возможно, удастся ускорить процесс вычисления, если сначала подынтегральную функцию приблизить с точностью ε полиномом $P_n(x)$ или рациональной функцией $w_{n,m}(x)$, а затем аналитически вычислить известным образом $\int_a^b P_n(x)\,dx$ или $\int_a^b w_{n,m}(x)\,dx$.

5.3.7. Суммирование рядов. Аналитические методы не ограничиваются только конечными вычислительными процессами, допускается предельный переход, а также суммирование бесконечных рядов. Например, интеграл от $y(x)$ можно заменить интегралом от ряда Тейлора

$$\int_a^b y(x)\,dx = \int_a^b \sum_{k=0}^{\infty} \frac{y^{(k)}(a)}{k!}(x-a)^k\,dx = \sum_{k=0}^{\infty} \frac{y^{(k)}(a)}{k!}\frac{(b-a)^{k+1}}{(k+1)} = \sum_{k=0}^{\infty} A_k;$$

тогда задача интегрирования сводится к задаче суммирования бесконечного числового ряда.

Представление решения в виде ряда имеет преимущество по сравнению с другими методами решения тогда, когда оценка отброшенной части ряда $\sum_{k=N+1}^{\infty} A_k$ легко определяется. Например, для знакопеременного ряда с монотонно убывающим $|A_k|$ погрешность вычисления $\varepsilon \leqslant |A_{N+1}|$. Чтобы вычислить

$$\int_0^1 e^{-x^2}\,dx = \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k+1)k!}$$

с точностью до восьми верных знаков, достаточно взять одиннадцать слагаемых в сумме

$$\sum_{k=0}^{11} \frac{(-1)^k}{(2k+1)k!}.$$

Так и следует поступать, если в библиотеке нет программы вычисления $\mathrm{erf}(x)$ с двойной точностью, поскольку одинарная точность не дает восьмой верный знак; следовательно, формулой из (5.3.2) воспользоваться нельзя.

Если ряд сходится быстро, то трудности при вычислении суммы ряда не возникают. Если же ряд сходится медленно, то суммирование ряда можно пытаться выполнить *методом Куммера*. Суть этого метода состоит в том, чтобы медленно сходящийся ряд $\sum_{k=0}^{\infty} A_k$ аппроксимировать рядом $\sum_{k=0}^{\infty} B_k$, для которого известна

сумма $S_1=\sum\limits_{k=0}^{\infty} B_k$, а скорость сходимости рядов одинакова, т. е. найти такие B_k, чтобы

$$\lim_{k\to\infty}\left|\frac{A_k-B_k}{A_k}\right|=0. \tag{5.3.3}$$

Тогда $S=\sum\limits_{k=0}^{\infty} A_k$ можно записать в виде $S=S_1+\sum\limits_{k=0}^{\infty}(A_k-B_k)$ и, таким образом, суммирование исходного ряда сводится к суммированию ряда с общим членом $C_k=(A_k-B_k)$, который, согласно (5.3.3), сходится быстрее исходного.

Предложенный подход можно повторить, если удается аппроксимировать ряд $\sum\limits_{k=0}^{\infty} C_k$ рядом с известной суммой и т. д.

Для ускорения сходимости рядов по методу Куммера необходимо иметь таблицу рядов с известными суммами. Подробные таблицы рядов представлены в [21—23].

Например, для медленно сходящегося ряда

$$S=\sum_{k=0}^{\infty}\frac{1-e^{-k^2}}{k^2+5^2}$$

можно для аппроксимации подобрать ряд с известной суммой

$$\sum_{k=0}^{\infty}\frac{1}{k^2+5^2}=\frac{1}{2\cdot 5^2}+\frac{\pi}{10}\operatorname{cth}5\pi=S_1.$$

Теперь имеем

$$S=S_1+\sum_{k=0}^{\infty}\frac{e^{-k^2}}{k^2+5^2}. \tag{5.3.4}$$

Заметим, что ряд в (5.3.4) является быстросходящимся, но и его сходимость можно ускорить методом Куммера. Запишем

$$\sum_{k=0}^{\infty}\frac{e^{-k^2}}{k^2+5^2}=\frac{1}{5^2}+\sum_{k=1}^{\infty}\frac{e^{-k^2}}{k^2}+\sum_{k=1}^{\infty}e^{-k^2}\left(\frac{1}{k^2+5^2}-\frac{1}{k^2}\right).$$

Ряд $S_2=\sum\limits_{k=1}^{\infty} e^{-k^2}k^{-2}$ имеет известную сумму $S_2=0{,}37247207$. Отсюда искомая сумма ряда представляется следующим образом:

$$S=S_1+0{,}04+S_2-\sum_{k=1}^{\infty}\frac{e^{-k^2}}{k^2((k/5)^2+1)},$$

где ряд сходится быстрее, чем в (5.3.4).

5.3.8. Фурье-анализ. Под аналитическим Фурье-анализом будем понимать: 1) аналитическое определение коэффициентов ряда Фурье по заданной аналитически (т. е. формулой) функции $f(x)$ периодической с периодом 2π, 2) аналитическое суммирование рядов Фурье, т. е. восстановление $f(x)$, заданной тригонометрическим рядом.

Фурье-анализ (аналитический и численный) широко применяется в науке и технике. Отметим две области приложений.

1. Разложение сложного колебания на отдельные гармонические колебания.

2. Решение задач математического анализа, дифференциальных уравнений методом Фурье, т. е. с помощью разложений в ряды Фурье.

Напомним, что коэффициенты Фурье вычисляются по формулам

$$a_m=\frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos mx\,dx,\ m=0,1,2,\ldots,$$
$$b_m=\frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin mx\,dx,\ m=1,2,\ldots, \tag{5.3.5}$$

а затем составляется ряд Фурье функции $f(x)$:

$$f(x)\sim\frac{a_0}{2}+\sum_{m=1}^{\infty}(a_m\cos mx+b_m\sin mx). \tag{5.3.6}$$

Для определения коэффициентов a_m, b_m достаточно абсолютной интегрируемости $f(x)$ на интервале $[-\pi,\pi]$. Для сходимости ряда (5.3.6) достаточно, чтобы $f(x)$ была кусочно-монотонна на $[-\pi,\pi]$ и имела конечное число точек разрыва. Тогда для точки непрерывности x_0

$$f(x_0)=\frac{a_0}{2}+\sum_{m=1}^{\infty}(a_m\cos mx_0+b_m\sin mx_0);$$

для точки разрыва

$$\frac{f(x_0+0)+f(x_0-0)}{2}=\frac{a_0}{2}+\sum_{m=1}^{\infty}(a_m\cos mx_0+b_m\sin mx_0).$$

Как отмечалось в п. 5.3.6, аналитическое интегрирование (5.3.5) связано с возможностью представить первообразную через известные элементарные или специальные функции. Этот вывод остается в силе и для определения коэффициентов Фурье a_m, b_m.

Приведем пример определения a_m, b_m для функции $e^{\alpha x}$, $-\pi<x<\pi$, $\alpha\neq 0$. Имеем:

$$a_0=\frac{1}{\pi}\int_{-\pi}^{\pi} e^{\alpha x}dx=\frac{e^{\alpha\pi}-e^{-\alpha\pi}}{\alpha\pi}=2\frac{\operatorname{sh}\alpha\pi}{\alpha\pi},$$

$$a_m=\frac{1}{\pi}\int_{-\pi}^{\pi} e^{\alpha x}\cos mx\,dx=\frac{1}{\pi}\frac{\alpha\cos mx+m\sin mx}{(\alpha^2+m^2)}e^{\alpha x}\Big|_{-\pi}^{\pi}=\frac{(-1)^m 2\alpha}{\pi(\alpha^2+m^2)}\operatorname{sh}\alpha\pi,$$

$$b_m=\frac{1}{\pi}\int_{-\pi}^{\pi} e^{\alpha x}\sin mx\,dx=\frac{1}{\pi}\frac{\alpha\sin mx-m\cos mx}{(\alpha^2+m^2)}e^{\alpha x}\Big|_{-\pi}^{\pi}=\frac{(-1)^{m-1}2m\operatorname{sh}\alpha\pi}{\pi(\alpha^2+m^2)}.$$

Таким образом, для $f(x)$, у которых интегралы в (5.3.5) можно вычислить аналитически, коэффициенты Фурье определяются вычислением известных функций, зависящих от номера гармоники m как от параметра.

Рассмотрим суммирование рядов Фурье. При этом будем считать, что факт сходимости ряда (5.3.6) установлен, а наша задача — найти в конечном виде сумму ряда, выразив ее через элементарные функции, если она в таком виде может быть представлена.

Изложим *метод Эйлера* суммирования некоторых тригонометрических рядов с помощью аналитических функций комплексной переменной.

Предположим, что имеем два ряда

$$\frac{1}{2}a_0 + \sum_{m=1}^{\infty} a_m \cos mx; \quad \sum_{m=1}^{\infty} a_m \sin mx,$$

которые всюду, за исключением конечного числа точек на интервале $[0,2\pi]$, сходятся: первый — к функции $p(x)$, второй — к $q(x)$.

Рассмотрим степенной ряд

$$\frac{1}{2}a_0 + \sum_{m=1}^{\infty} a_m z^m, \tag{5.3.7}$$

где z — комплексная переменная.

На окружности $|z|=1$, $z=e^{ix}$ этот ряд сходится по предположению, за исключением конечного числа точек, т. е.

$$p(x)+iq(x)=\frac{1}{2}a_0 + \sum_{m=1}^{\infty} a_m z^m =$$

$$=\frac{1}{2}a_0 + \sum_{m+1}^{\infty} a_m(\cos mx + i\sin mx). \tag{5.3.8}$$

Но, согласно известному свойству степенных рядов (5.3.7) сходится при $|z|<1$, $z=\rho e^{ix}$, $0\leqslant\rho<1$, к некоторой функции $\varphi(z)$. Тогда имеет место

$$\varphi(\rho e^{ix})=\frac{1}{2}a_0 + \sum_{m=1}^{\infty} a_m \rho^m e^{imx}.$$

Если ряд (5.3.8) сходится, то по теореме Абеля его сумма находится предельным переходом

$$\lim_{\rho\to 1} \varphi(\rho e^{ix})=p(x)+iq(x).$$

Как правило, $\lim\limits_{\rho\to 1} \varphi(\rho e^{ix})=\varphi(e^{ix})$, а отсюда уже $p(x)$, $q(x)$ можно получить в конечном виде.

Просуммируем методом Эйлера два ряда:

$$p(x)=1+\sum_{m=1}^{\infty}\frac{1}{m!}\cos mx; \quad q(x)=\sum_{m=1}^{\infty}\frac{1}{m!}\sin mx.$$

Очевидно, что

$$\varphi(z)=1+\sum_{m=1}^{\infty}\frac{z^m}{m!}=e^z.$$

Отсюда

$$\varphi(e^{ix})=e^{e^{ix}}=e^{\cos x+i\sin x}=e^{\cos x}(\cos(\sin x)+i\sin(\sin x)).$$

Следовательно,

$$p(x)=e^{\cos x}\cos(\sin x),\quad q(x)=e^{\cos x}\sin(\sin x).$$

5.3.9. Дифференцирование. Процедура аналитического дифференцирования функций одного или нескольких переменных, заданных формулами, является основной во многих вычислительных методах.

Построение рядов Тейлора в окрестности точки

$$f(x)=f(x_0)+\frac{f'(x_0)}{1!}(x-x_0)+\frac{f''(x_0)}{2!}(x-x_0)^2+\ldots$$

требует вычисления $f^{(n)}(x)$ в точке x_0. Поиск экстремумов функции $f(x_1, \ldots, x_n)$ может быть связан с решением уравнений

$$\frac{\partial f}{\partial x_i}(x_1, \ldots, x_n)=0$$

и проверкой знакоопределенности d^2f. Вычисление преобразования $y_i=f_i(x_1, \ldots, x_n)$, $1\leqslant i\leqslant n$,

$$\det\left|\frac{\partial y_i}{\partial x_j}(x_1^0, \ldots, x_n^0)\right|=\frac{D(y_1, \ldots, y_n)}{D(x_1, \ldots, x_n)}$$

часто необходимо при заменах переменных. В задачах дифференциальной геометрии — определение касательных поверхностей, вычисление кривизны поверхности и т. п.— аналитическое дифференцирование служит основным элементом вычислений. В механике вывод уравнения движения связан с дифференцированием функции Лагранжа $L=L(t, y_i, y_i')$, $1\leqslant i\leqslant n$, где y_i, y_i' — обобщенные координаты и скорости механической системы. Действительно, уравнения Лагранжа имеют вид

$$\frac{d}{dt}\left(\frac{\partial L}{\partial y_i'}\right)-\frac{\partial L}{\partial y_i}=0,\quad 1\leqslant i\leqslant n;$$

следовательно, для получения уравнений движения в виде

$$y_i''=\varphi_i(y_i', y_i, t),\quad 1\leqslant i\leqslant n,$$

необходимо провести аналитическое дифференцирование функции L. Заметим, что при больших значениях n эта задача становится очень трудоемкой. Именно в таких задачах при больших n используют системы аналитических вычислений на ЭВМ.

Для аналитического дифференцирования необходимо знать таблицу производных элементарных функций и правила дифференцирования. Все это входит в курс высшей математики.

5.3.10. Операции с матрицами и векторами. Конечное число матричных и векторных операций представляет собой простые комбинации элементарных арифметических операций. Например, сумма двух прямоугольных матриц

$$A=(a_{i,j}),\ B=(b_{i,j}),\ C=(c_{i,j}),\ 1\leqslant i\leqslant m,\ 1\leqslant j\leqslant n,$$

$$C=A+B,\ c_{i,j}=a_{i,j}+b_{i,j};$$

произведение двух прямоугольных матриц

$$A=(a_{i,j}),\ B=(b_{i,j}),\ 1\leqslant i\leqslant m,\ 1\leqslant j\leqslant n,\ 1\leqslant i\leqslant n,\ 1\leqslant j\leqslant k,$$

$$C=AB,\ c_{i,j}=\sum_{s=1}^{n} a_{i,s}b_{s,j},\ 1\leqslant i\leqslant m,\ 1\leqslant j\leqslant k;$$

произведение квадратной матрицы $A=(a_{i,j})$, $1\leqslant i,j\leqslant n$, на вектор $x=(x_i)$, $1\leqslant i\leqslant n$,

$$y=Ax,\ y_i=\sum_{j=1}^{n} a_{i,j}x_j$$

и т. д. Исключением является операция обращения квадратной матрицы A с $\det A\neq 0$, аналитические методы выполнения которой рассматриваются в п. 5.3.11.

Если элементы матриц и векторов — числа, то конечное число операций с матрицами и векторами следует отнести к численным методам. Если же допускается бесконечное число операций, то в этом случае могут применяться аналитические методы. Важным классом бесконечных процессов с матрицами является процедура определения функций от матриц рядами.

Для определения сходимости последовательности матриц и матричных рядов напомним понятие нормы.

Пусть имеется n-мерное пространство E^n. Если для любого вектора $x\in E^n$ существует число $\|x\|$ такое, что:

1) $\|\alpha x\|=|\alpha|\,\|x\|$ для любого числа x;

2) $\|x+y\|\leqslant\|x\|+\|y\|$ для любых $x,\ y\in E^n$;

3) $\|x\|=0$, если $x=0$,

то $\|x\|$ называется *нормой* вектора $x=(x,\ ...,\ x_n)$. Например, нормой вектора x являются следующие:

а) евклидова норма $\|x\|_2=\left(\sum_{i=1}^{n} x_i^2\right)^{1/2}$;

б) $\|x\|_1=\sum_{i=1}^{n}|x_i|$;

в) $\|x\|_\infty=\max\limits_{1\leqslant i\leqslant n}(|x_i|)$.

Пусть A — матрица, x — любой вектор E^n, $\|x\|\neq 0$, нормой матрицы $\|A\|$ называется наименьшее из чисел c в неравенстве

$$\|Ax\|\leqslant c\,\|x\|.$$

В зависимости от применяемой нормы вектора будут получаться различные значения нормы матрицы. Из определения $\| A \|$ имеем

$$\| Ax \| \leqslant \| A \| \, \| x \|.$$

В дальнейшем будем применять норму вектора $\| x \|_\infty$ и опускать индекс ∞:

$$\| x \| = \max_{1 \leqslant i \leqslant n} (| x_i |).$$

Соответствующая этой норме матричная норма $\| A \|$ может вычисляться по элементам $a_{i,j}$ следующим образом:

$$\| A \| = \max_{1 \leqslant i \leqslant n} \left(\sum_{j=1}^{n} | a_{i,j} | \right).$$

Определим последовательность матриц $\{S^{(k)}\}$, задавая $n \times n$ числовых последовательностей

$$\{S_{i,j}^{(k)}\}, \quad 1 \leqslant i, j \leqslant n, \quad k = 1, 2, 3, \ldots .$$

Последовательность $\{S^{(k)}\}$ сходится к матрице S с элементами $S_{i,j}$, если сходятся $n \times n$ числовых последовательностей:

$$S_{i,j}^{(k)} \to S_{i,j} \text{ при } k \to \infty$$

или

$$\| S^{(k)} - S \| \to 0 \text{ при } k \to \infty.$$

Напомним определение собственных чисел λ_i и собственных векторов e_i матрицы A. Собственные числа $\lambda_i(A)$ матрицы A и соответствующие векторы удовлетворяют равенству

$$Ae_i = \lambda_i(A) e_i, \quad 1 \leqslant i \leqslant m, \quad e_i \neq 0.$$

Пусть функция $f(\lambda)$ разлагается в степенной ряд в круге сходимости $|\lambda - \lambda_0| < r$:

$$f(\lambda) = \sum_{i=0}^{\infty} a_i (\lambda - \lambda_0)^i,$$

где a_i — числовые коэффициенты (возможно, и комплексные). Сопоставим этому ряду последовательность частичных сумм — матричную последовательность

$$S^{(k)} = \sum_{i=0}^{k} a_i (A - \lambda_0 E)^i. \tag{5.3.9}$$

Теорема 5.1. *Для сходимости $S^{(k)} \to S$ необходимо и достаточно, чтобы все собственные значения $\lambda_i(A)$ лежали внутри круга сходимости $|\lambda_i(A) - \lambda_0| < r$.*

Пусть выполнено условие теоремы 5.1; тогда $S^{(k)} \to S$, т. е. сходится матричный ряд

$$S = \sum_{i=0}^{\infty} a_i (A - \lambda_0 E)^i;$$

его сумму (матрицу S) обозначают $f(A)$,

$$f(A)=\sum_{i=0}^{\infty} a_i(A-\lambda_0 E)^i.$$

Так по функции $f(x)$ определяется функция f от матрицы A, которая обозначается $f(A)$.

Для применения теоремы 5.1 важно уметь оценивать расположение собственных значений $\lambda_i(A)$ на комплексной плоскости. Простейшую оценку можно получить из следующей теоремы.

Теорема 5.2. *Собственные числа* $\lambda_i(A)$ *имеют оценку*

$$\max_i |\lambda_i(A)| \leqslant \| A \|. \tag{5.3.10}$$

Доказательство. Из определения собственных значений и векторов имеем

$$Ae_i=\lambda_i(A)\,e_i.$$

Отсюда для любого i

$$\| Ae_i \| = \| \lambda_i(A)\,e_i \| = |\lambda_i(A)|\,\| e_i \|.$$

С другой стороны, по определению нормы матрицы,

$$\| Ae_i \| \leqslant \| A \|\,\| e_i \|.$$

Сравнивая два последних соотношения, получаем (5.3.10).

Например, в круге сходимости $|\lambda|<1$ имеет место разложение

$$f(\lambda)=(1-\lambda)^{-1}=\sum_{i=0}^{\infty}\lambda.$$

Пусть имеем матрицу A

$$A=\begin{pmatrix} 0{,}3 & -0{,}1 \\ -0{,}1 & 0{,}2 \end{pmatrix}.$$

Определим $\| A \|=0{,}4$. Используя (5.3.10), находим

$$\max |\lambda_i(A)| \leqslant 0{,}4.$$

По теореме 5.1 матричный ряд $\sum\limits_{i=1}^{\infty} A^i$ сходится к сумме — матрице $(E-A)^{-1}$:

$$(E-A)^{-1}=\sum_{i=0}^{\infty} A^i.$$

Если радиус сходимости для $f(\lambda)$ равен бесконечности, то можно получить функцию $f(A)$, определенную для любых матриц A. Имеем

$$e^A=\sum_{i=0}^{\infty}\frac{1}{i!}A^i;\quad \cos A=\sum_{i=0}^{\infty}\frac{(-1)^i}{(2i)!}A^{2i};\quad \operatorname{sh} A=\sum_{i=0}^{\infty}\frac{1}{(2i+1)!}A^{2i+1}.$$

5.3.11. Решение систем линейных уравнений. Как уже упоминалось в п. 5.1.3, аналитическим методом решения систем линейных уравнений

$$Ax=b, \tag{5.3.11}$$

где $A=(a_{i,j})$, $1\leqslant i,j\leqslant n$, $b=(b_1, ..., b_n)$, x — вектор неизвестных $x=(x_1, ..., x_n)$ с определителем $\det A\neq 0$, является правило Крамера. Однако для больших n оно неприменимо, так как становится громоздким.

Другим аналитическим методом решения системы (5.3.11) является формула Фробениуса для обращения блочных матриц.

Определим блочную матрицу. Пусть дана прямоугольная матрица

$$A=(a_{i,j}),\ 1\leqslant i\leqslant m,\ 1\leqslant j\leqslant n.$$

Представим ту же самую матрицу в виде

$$A=(A_{q,r}),\ 1\leqslant q\leqslant l,\ 1\leqslant r\leqslant s,$$

где каждый элемент $A_{q,r}$ может быть прямоугольной матрицей. Например,

$$A=\left(\begin{array}{cc|ccc} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} \\ \hline a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & a_{4,5} \end{array}\right)=\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix}, \tag{5.3.12}$$

где

$$A_{1,1}=\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix},\quad A_{1,2}=\begin{pmatrix} a_{1,3} & a_{1,4} & a_{1,5} \\ a_{2,3} & a_{2,4} & a_{2,5} \end{pmatrix},$$

$$A_{2,1}=\begin{pmatrix} a_{3,1} & a_{3,2} \\ a_{4,1} & a_{4,2} \end{pmatrix},\quad A_{2,2}=\begin{pmatrix} a_{3,3} & a_{3,4} & a_{3,5} \\ a_{4,3} & a_{4,4} & a_{4,5} \end{pmatrix}.$$

Действия над блочными матрицами производятся по тем же правилам, как и в случае, когда вместо блоков имеются числа. Необходимо только, чтобы разбиение на блоки было одинаковым. Пусть имеем матрицу $B=(b_{i,j})$, $1\leqslant i\leqslant m$, $1\leqslant j\leqslant n$, разбитую на блоки тех же размеров, что и A, т. е.

$$B=(B_{q,r}),\ 1\leqslant q\leqslant l,\ 1\leqslant r\leqslant s.$$

Сумма C в блочном представлении получается сложением соответствующих блоков

$$C=A+B=(A_{q,r}+B_{q,r}).$$

Для умножения блочных матриц A и B необходимо, чтобы все горизонтальные размеры в разбиении A совпадали с соответствую-

щими вертикальными размерами в разбиении B. Тогда формула для блочного представления

$$C=(C_{q,r})=\sum_{k=1}^{s} A_{q,k}B_{k,r}$$

аналогична формуле для матриц, состоящих из чисел. Пусть имеем матрицу

$$B=\left(\begin{array}{ccc|cc} b_{1,1} & b_{1,2} & b_{1,3} & b_{1,4} & b_{1,5} \\ b_{2,1} & b_{2,2} & b_{2,3} & b_{2,4} & b_{2,5} \\ \hline b_{3,1} & b_{3,2} & b_{3,3} & b_{3,4} & b_{3,5} \\ b_{4,1} & b_{4,2} & b_{4,3} & b_{4,4} & b_{4,5} \\ b_{5,1} & b_{5,2} & b_{5,3} & b_{5,4} & b_{5,5} \end{array}\right)=\begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix}, \qquad (5.3.13)$$

где

$$B_{1,1}=\begin{pmatrix} b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,1} & b_{2,2} & b_{2,3} \end{pmatrix},\quad B_{1,2}=\begin{pmatrix} b_{1,4} & b_{1,5} \\ b_{2,4} & b_{2,5} \end{pmatrix},$$

$$B_{2,1}=\begin{pmatrix} b_{3,1} & b_{3,2} & b_{3,3} \\ b_{4,1} & b_{4,2} & b_{4,3} \\ b_{5,1} & b_{5,2} & b_{5,3} \end{pmatrix},\quad B_{2,2}=\begin{pmatrix} b_{3,4} & b_{3,5} \\ b_{4,4} & b_{4,5} \\ b_{5,4} & b_{5,5} \end{pmatrix}.$$

При таком разбиении матриц A, B блочное представление матрицы $C=AB$, равное произведению (5.3.12) и (5.3.13), находится по формулам

$$C=\begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}=\left(\begin{array}{c|c} A_{1,1}B_{1,1}+A_{1,2}B_{2,1} & \cdots \\ \hline A_{2,1}B_{1,1}+A_{2,2}B_{2,1} & \cdots \end{array}\right).$$

Представим формулу Фробениуса обращения блочной матрицы. Пусть в (5.3.11) матрица A — квадратная, $\det A\neq 0$ и разбита на блоки

$$A=\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix};$$

матрица $A_{1,1}$ — квадратная, $\det A_{1,1}\neq 0$. Определим матрицу

$$B=A_{2,2}-A_{2,1}(A_{1,1})^{-1}A_{1,2}.$$

Предположим, что $\det B\neq 0$. Тогда блочное представление обратной матрицы A^{-1} задается *формулой Фробениуса*

$$A^{-1}=\left(\begin{array}{c|c} (A_{1,1})^{-1}+(A_{1,1})^{-1}A_{1,2}B^{-1}A_{2,1}(A_{1,1})^{-1} & -(A_{1,1})^{-1}AB^{-1} \\ \hline -B^{-1}A_{2,1}(A_{1,1})^{-1} & B^{-1} \end{array}\right),$$

которую можно проверить прямым умножением матриц A и A^{-1} в блочном виде.

Решение исходной системы уравнений (5.3.11) получается умножением блочной матрицы A^{-1} на вектор правых частей, разбитый на блоки в соответствии с разбиением (A^{-1}).

Главное достоинство формулы Фробениуса состоит в том, что обращение матрицы размерности n сводится к двум обращениям матриц $A_{1,1}$ и B меньшей размерности, а это может позволить найти аналитическое решение (5.3.11).

Можно продолжить процедуру обращения Фробениуса, разбивая на блоки матрицы $A_{1,1}$ и B и т. д., но, как обычно, с каждым шагом увеличиваются трудности, поскольку приходится манипулировать громоздкими формулами.

Например, матрицу 16-го порядка за три шага можно свести к обращению матриц 2-го порядка, для которых легко применить правило Крамера и, таким образом, получить при условии выполнимости этих шагов аналитическое решение системы линейных уравнений (5.3.11), выраженное через элементы матрицы A и вектора b.

Как отмечалось выше, решая систему (5.3.11), необходимо знать, обращается или нет в нуль определитель матрицы A, $A_{1,1}$, B. Матрица A называется *вырожденной*, если определитель $\det A=0$, и *невырожденной*, если $\det A\neq 0$.

Иногда можно по элементам матрицы легко определить, что матрица A невырождена. Это случай диагонального преобладания Адамара.

Теорема 5.3. *Если для матрицы A выполняются n неравенств*

$$|a_{i,i}|>\sum_{\substack{j=1\\ j\neq i}}^{n}|a_{i,j}|,\quad 1\leqslant i\leqslant n, \tag{5.3.14}$$

то матрица A невырождена.

Доказательство. Допустим противное: при выполнении (5.3.14) $\det A=0$. Тогда существует ненулевой вектор $x=(x_1, ..., x_n)$ такой, что

$$\sum_{j=1}^{n}a_{i,j}x_j=0,\quad 1\leqslant i\leqslant n.$$

Выберем число k, соответствующее $\max\limits_{1\leqslant i\leqslant n}|x_i|=|x_k|>0$. Из последнего равенства для выбранного k получаем

$$a_{k,k}x_k=-\sum_{\substack{j=1\\ j\neq k}}^{n}a_{k,j}x_j,$$

откуда

$$|a_{k,k}||x_k|\leqslant\sum_{\substack{j=1\\ j\neq k}}^{n}|a_{k,j}||x_j|\leqslant|x_k|\sum_{\substack{j=1\\ j\neq k}}^{n}|a_{k,j}|.$$

Сокращая на $|x_k|$, получим противоречие с (5.3.14). Теорема доказана.

Обозначим A' транспонированную матрицу к A, $A'=(a_{j,i})$, $1\leqslant j, i\leqslant n$. Так как

$$\det A=\det A',$$

то достаточные условия Адамара (5.3.14) для строк, заменяя A на A', могут быть преобразованы в условия для столбцов. Матрица A невырождена, если

$$|a_{i,i}|>\sum_{\substack{j=1\\ j\neq i}}^{n}|a_{j,i}|,\quad 1\leqslant i\leqslant n.$$

Условия Адамара перенумерацией строк (столбцов) можно сформулировать в виде $\det A\neq 0$, если в каждой строке (столбце) A имеется преобладающий элемент и эти элементы расположены в различных столбцах (строках).

Пример. $\det A\neq 0$ для следующей матрицы:

$$A=\begin{pmatrix} 15 & 1 & 2 & -4 \\ 2 & 3 & -10 & 4 \\ -1 & 7 & 0 & 1 \\ 1 & 1 & 2 & 10 \end{pmatrix}\quad\begin{aligned} 15&>1+2+4,\\ 10&>2+3+4,\\ 7&>1+0+1,\\ 10&>1+1+2. \end{aligned}$$

Те же элементы (15, -10, 7, 10) являются преобладающими и по столбцам.

5.3.12. Определение собственных значений и векторов. Аналитическое определение $\lambda_i(A)$ собственных значений матрицы A и соответствующих собственных векторов e_i в конечном виде для произвольных матриц A порядка $n\geqslant 5$ невозможно. Это следует из того факта, что λ_i удовлетворяют алгебраическому уравнению n-й степени

$$\det(A-\lambda E)=0,$$

которое в развернутой записи имеет вид

$$\lambda^n+a_1\lambda^{n-1}+a_2\lambda^{n-2}+\ldots+a_{n-1}\lambda+a_n=0, \qquad (5.3.15)$$

а (5.3.15), как уже упоминалось, в радикалах неразрешимо при $n\geqslant 5$.

Поэтому аналитические методы нахождения собственных значений и векторов, т. е. их определение через элементы матрицы A, существенно зависят от n.

Для $1\leqslant n\leqslant 4$ можно использовать известные формулы для корней λ_i алгебраических уравнений, а затем аналитически решить систему линейных уравнений

$$Ae_i=\lambda_i e_i \qquad (5.3.16)$$

и определить соответствующие собственные векторы.

Для $n\geqslant 5$ аналитические решения могут дать методы возмущений (см. **5.4**).

Однако если отказаться от желания найти точные формулы для λ_i и ограничиться поиском их оценок или, как говорят, решать задачу локализации собственных значений, то здесь имеются простые аналитические методы для произвольных порядков матрицы A. Часто в технических задачах достаточно иметь именно оценку для λ_i, а не точные значения. Например, в задачах устойчивости дифференциальных уравнений, систем

автоматического управления определяются знаки $\mathrm{Re}\lambda_i$, а отсюда делается заключение об устойчивости без точного определения $\lambda_i(A)$.

Критерии локализации собственных значений могут применяться и для задачи локализации корней полиномов, поскольку по любому алгебраическому уравнению (5.3.15) можно выписать матрицу, для которой это уравнение является характеристическим, т. е. $\det(A-\lambda E)=0$, а именно:

$$A=\begin{pmatrix} 0 & 1 & 0\ldots & 0 & 0 \\ 0 & 0 & 1\ldots & 0 & 0 \\ \ldots. & \ldots. & \ldots\ldots\ldots & \ldots. & \ldots. \\ 0 & 0 & 0\ldots & 0 & 1 \\ -a_n & -a_{n-1} & -a_{n-2} & -a_2 & -a_1 \end{pmatrix}.$$

Первым критерием локализации собственных значений является формула (5.3.10), которая дает оценку модулей собственных чисел. Второй критерий является следствием теоремы 5.3.3.

Все собственные числа матрицы A содержатся в объединении кругов

$$|\lambda-a_{i,i}|\leqslant\sum_{j=1}^{n}|a_{ij}|,\quad 1\leqslant i\leqslant n,\quad i\neq j,$$

или

$$|\lambda-a_{i,i}|\leqslant\sum_{j=1}^{n}|a_{j,i}|,\quad 1\leqslant i\leqslant n,\quad i\neq j,$$

на комплексной λ-плоскости.

Докажем первое неравенство. Пусть оно не выполняется для какого-либо собственного числа λ_* и всех $1\leqslant i\leqslant n$; тогда

$$|\lambda_*-a_{i,i}|>\sum_{\substack{j=1\\ i\neq j}}^{n}|a_{i,j}|,\quad 1\leqslant i\leqslant n;$$

следовательно, по теореме 5.3, матрица

$$A-\lambda_* E$$

невырождена, т. е. $\det(A-\lambda_* E)\neq 0$, а это противоречит предположению, что λ_* — собственное число матрицы A.

5.3.13. Решение линейных интегральных уравнений. В этом пункте представлены некоторые типы интегральных уравнений, которые можно решать аналитически. Это значит, что можно: 1) представить явную формулу для решения в виде интегралов от заданных функций или их преобразований; 2) свести задачу к решению системы линейных алгебраических уравнений и интегрированию заданных функций; 3) представить решение сходящимся рядом, каждый член которого определяется интегрированием заданных функций.

Рассматриваемые интегральные уравнения можно записать следующим образом:

$$ay(x)+\int\limits_{D_1} K(x, s)\,y(s)\,ds=f(x),\quad x\in D,$$

где $y(x)$ — искомая функция, стоящая под знаком интеграла, отсюда и название *интегральное уравнение*; $f(x)$, $K(x, s)$ — заданные функции; $f(x)$ — неоднородность уравнения; $K(x, s)$ — ядро уравнения; a — константа, если $a=0$, то имеем уравнение первого рода, если $a\neq 0$ — второго рода; D_1 — область интегрирования по переменной s; D — область изменения аргумента x.

Обобщением рядов Фурье на непериодические функции являются: интегральное преобразование Фурье — косинус-преобразование функции $y(x)$

$$\sqrt{\frac{2}{\pi}}\int\limits_0^{\infty}(\cos xs)\,y(s)\,ds=f(x),$$

синус-преобразование функции $y(x)$

$$\sqrt{\frac{2}{\pi}}\int\limits_0^{+\infty}(\sin xs)\,y(s)\,ds=f(x),$$

комплексное преобразование $y(x)$

$$\frac{1}{\sqrt{2\pi}}\int\limits_{\infty}^{+\infty}\mathrm{e}^{-isx}y(s)\,ds=f(x).$$

Эти интегральные преобразования при заданной функции $f(x)$ можно трактовать как интегральные уравнения первого рода относительно неизвестной функции $y(x)$.

Если $f(x)$ абсолютно интегрируема в каждом конечном промежутке и при $x\to\pm\infty$ монотонно стремится к нулю, то можно применить простые формулы обращения: для косинус-преобразования

$$y(x)=\sqrt{\frac{2}{\pi}}\int\limits_0^{\infty}f(s)\cos(sx)\,ds,$$

для синус-преобразования

$$y(x)=\sqrt{\frac{2}{\pi}}\int\limits_0^{\infty}f(s)\sin(sx)\,ds,$$

для комплексного преобразования

$$y(x)=\frac{1}{\sqrt{2\pi}}\int\limits_{-\infty}^{+\infty}\mathrm{e}^{isx}f(s)\,ds.$$

Рассмотрим уравнение, в котором $K(x, s)$ зависит от разности аргументов: $K(x, s)=K(x-s)$ — интегральное уравнение с разностным ядром $K(x-s)$

$$ay(x)+\int_{-\infty}^{+\infty} K(x-s)\,y(s)\,ds=f(x). \tag{5.3.17}$$

Интеграл $\int_{-\infty}^{+\infty} K(x-s)\,y(s)\,ds$ называется *сверткой функций* K и y. Применим к обеим частям уравнения комплексное преобразование Фурье; допуская, что применима теорема об умножении Фурье-образа свертки [29], находим

$$\left[a-\sqrt{2\pi}\,\tilde{K}(x)\right]\tilde{y}(x)=\tilde{f}(x).$$

Здесь знаком $\sim$ обозначено преобразование Фурье функции. Предположим, что функция $\left[a-\sqrt{2\pi}\,\tilde{K}(x)\right]^{-1}$ ограничена на всей оси $-\infty<x<+\infty$. Тогда получим следующую формулу для решения (5.3.17):

$$y(x)=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{+\infty}\frac{e^{isx}\tilde{f}(x)}{a-\sqrt{2\pi}\,\tilde{K}(s)}\,ds.$$

Уравнения вида (5.3.17) часто имеют место в задачах определения сигнала $y(x)$, подверженного преобразованию (ядро $K(x-s)$), по экспериментально наблюдаемой функции $f(x)$ (измерения).

Другим интегральным уравнением с известным аналитическим решением является уравнение Абеля

$$\int_a^x \frac{y(s)}{(x-s)^{\alpha}}\,ds=f(x),\quad 0<\alpha<1. \tag{5.3.18}$$

Если $f(x)$ непрерывно дифференцируема при $x\geqslant a$, то решение уравнения Абеля можно записать в виде

$$y(x)=\frac{\sin\alpha\pi}{\pi}\,\frac{d}{dx}\int_a^x\frac{f(s)}{(x-s)^{1-\alpha}}\,ds.$$

Эта формула получается заменой x на s умножением уравнения (5.3.18) на $(x-s)^{\alpha-1}$ и интегрированием по s от a до x. При этом используем формулу

$$\int_0^1\frac{ds}{s^{\alpha}(1-s)^{1-\alpha}}=\frac{\pi}{\sin\alpha\pi}.$$

Аналитическое решение следующих интегральных уравнений, которые называются *вырожденными*, можно свести к аналитичес-

кому решению системы линейных алгебраических уравнений. Вырожденные уравнения

$$ay(x)+\int_D K(x, s)\,y(s)\,ds=f(x),\quad x\in D, \tag{5.3.19}$$

имеют ядро $K(x, s)$, представимое в виде

$$K(x, s)=\sum_{k=1}^{n} a_k(x)\,b_k(s).$$

Функции $a_k(x)$, $b_k(s)$ линейно независимы, $a\neq 0$. Определим постоянные c_k:

$$c_k=\int_{D_1} b_k(s)\,y(s)\,dx,\quad k=1, 2, \ldots, n.$$

Тогда исходное интегральное уравнение запишется в виде

$$ay(x)+\sum_{k=1}^{n} c_k a_k(x)=f(x).$$

Умножая это соотношение последовательно на $b_i(x)$ и интегрируя по области D, получим систему линейных алгебраических уравнений относительно c_j:

$$ac_j+\sum_{k=1}^{n} A_{k,j}c_k=d_j,\quad 1\leqslant j\leqslant n, \tag{5.3.20}$$

где

$$A_{k,j}=\int_D a_k(x)\,b_j(x)\,dx,\quad d_j=\int_D f(x)\,b_j(x)\,dx.$$

Если матрица $A_{k,j}$ и вектор d_j могут быть определены аналитически, а затем (5.3.20) также решается аналитически относительно c_j, то решение интегрального уравнения находится по формуле

$$y(x)=\frac{1}{a}\left(f(x)-\sum_{k=1}^{n} a_k(x)\,c_k\right).$$

Наконец, рассмотрим метод последовательных приближений решения (5.3.19), который при определенных условиях может дать аналитическое решение. Построим по (5.3.19) последовательность функций $y_k(x)$ таким образом:

$$y_k(x)=\frac{1}{a}\left(f(x)-\int_{D_1} K(x, s)\,y_{k-1}(s)\,ds\right), \tag{5.3.21}$$

$k=1, 2, \ldots;\ y_0(x)\equiv 0$. Если последовательность функций $\{y_k(x)\}$ имеет предел $y(x)$ в некотором функциональном пространстве, возможен предельный переход под знаком интеграла в (5.3.21)

и при любом k интегралы в (5.3.21) можно взять аналитически, то формула

$$y(x)=\lim_{k\to\infty} y_k(x)$$

дает аналитическое решение (5.3.19).

5.3.14. Корни полиномов. Как уже отмечалось, точное аналитическое решение алгебраического уравнения

$$P_n(x)=a_0x^n+a_1x^{n-1}+\ldots+a_{n-1}x+a_n=0$$

для произвольного полинома $P_n(x)$ возможно при $n\leqslant 4$. Однако для многих приложений, связанных с исследованием устойчивости динамических систем, важно показать, что корни $P_n(x)$ имеют отрицательные вещественные части, а не определять их точные значения. В данном пункте рассматриваются необходимые и достаточные условия для того, чтобы полином $P_n(x)$ с вещественными коэффициентами имел корни с отрицательными вещественными частями.

Обозначим корни $P_n(x)$: вещественные x_i, $1\leqslant i\leqslant k$; комплексные $z_j=\alpha_j+i\beta_j$, $1\leqslant j\leqslant(n-k)/2$.

Теорема 5.4. *Для того чтобы выполнялись неравенства* $x_i<0$, $\alpha_j<0$ *необходимо, чтобы все коэффициенты* a_i $(1\leqslant i\leqslant n)$ *имели тот же знак, что и* a_0.

Доказательство. Представим $P_n(x)$ разложением на множители следующим образом:

$$\frac{1}{a_0}P_n(x)=(x-x_1)\ldots(x-x_k)(x-\alpha_1-i\beta_1)(x-\alpha_1+i\beta_1)\ldots$$

$$\ldots(x-\alpha_{(n-k)/2}-i\beta_{(n-k)/2})(x-\alpha_{(n-k)/2}+i\beta_{(n-k)/2})=(x-x_1)\ldots$$

$$\ldots(x-x_k)(x^2-2\alpha_1x+\alpha_1^2+\beta_1^2)\ldots(x^2-2\alpha_{(n-k)/2}x+\alpha_{(n-k)/2}^2+\beta_{(n-k)/2}^2).$$

Поскольку $x_i<0$, $\alpha_j<0$, в правой части знаки при степенях x строго положительны, а следовательно, все коэффициенты в $P_n(x)$ имеют знак, совпадающий со знаком a_0, что и требовалось доказать.

Введем определители Гурвица

$$\Delta_1=a_1,\ \Delta_2=\begin{vmatrix}a_1 & a_3\\ a_0 & a_2\end{vmatrix},\ \ldots,\ \Delta_n=\begin{vmatrix}a_1 & a_3 & a_5 & \ldots\\ a_0 & a_2 & a_4 & \ldots\\ 0 & a_1 & a_3 & \ldots\\ 0 & a_0 & a_2 & \ldots\\ \ldots & \ldots & \ldots & \ldots\\ \ldots & \ldots & \ldots & a_n\end{vmatrix}$$

Справедлив критерий: *для того чтобы полином* $P_n(x)$ *при* $a_0>0$ *имел все корни с отрицательными вещественными частями, необходимо и достаточно*, чтобы:

1) $a_i>0$, $1\leqslant i\leqslant n$;
2) $\Delta_{n-1}>0$, $\Delta_{n-3}>0$, ...

5.3.15. Корни нелинейных уравнений. Элементарные трансцендентные функции ($\sin x$, $\operatorname{tg}x$ и т. п.) имеют на вещественной оси

$-\infty < x < +\infty$ бесконечное число нулей. Однако если ограничиться конечной областью D изменения независимого переменного x и заданной точностью вычислений корней, то при достаточной гладкости функции $f(x)$ в D можно для отыскания корней уравнения

$$f(x)=0 \tag{5.3.22}$$

использовать аналитические методы определения корней полинома.

Заменим (5.3.22) отрезком ряда Тейлора с центром в x_0 с остаточным членом в форме Лагранжа

$$f(x)=f(x_0)+\frac{f'x_0}{1!}(x-x_0)+\frac{f''(x_0)}{2!}(x-x_0)^2+\ldots+\frac{f^n(x_0)}{n!}(x-x_0)^n+$$

$$+\frac{f^{n+1}(c(x))}{(n+1)!}(x-x_0)^{n-1}, \tag{5.3.23}$$

где $c=c(x)$ — точка между x и x_0. Затем в (5.3.23) следует отбросить остаточный член, найти аналитически корни полинома n-й степени, оценить ошибку в определении корней, связанную с заменой $f(x)$ на $P_n(x)$, т. е. показать, что удовлетворяется заданная точность вычисления корней в области D.

Например, если уравнение $\cos x=0$ в области $0 \leqslant x \leqslant 5$ заменить уравнением $P_4(x)=0$

$$1-x^2/2+x^4/24=0$$

(отрезок ряда Тейлора с центром в точке $x_0=0$), то точный первый корень $-x_{1,*}=\pi/2$ определяется формулой

$$x_{1,2}=\sqrt{6\pm\sqrt{36-24}}=\sqrt{6\pm 2\sqrt{3}}$$

с ошибкой $\sim 0{,}02$, второй — $x_{2,*}=\frac{3}{2}\pi$ с ошибкой ~ 2.

Если разложить $\cos x$ в ряд Тейлора с центром в точке $x_0=\pi$ и оставить слагаемые до четвертой степени включительно, то получим уравнение

$$-1+\frac{(x-\pi)^2}{2}-\frac{1}{24}(x-\pi)^4=0,$$

откуда первый и второй корень определяются с погрешностью $\sim 0{,}02$:

$$x_{1,2}=\pi\pm\sqrt{6-2\sqrt{3}}.$$

Как и при аппроксимации функций, полином $P_n(x)$ может выбираться не только на основе ряда Тейлора, но и любым другим способом. Важно только уметь оценивать погрешность замены $f(x)$ на $P_n(x)$ — соответствующую погрешность определения корней. Кроме того, можно заменять $f(x)$ дробно-рациональной функцией $w_{n,m}(x)$ и искать нули числителя.

5.3.16. Нелинейная оптимизация. Поиск минимума или максимума нелинейной функции $f(x)$ в некоторой области D — простейшая

задача нелинейной оптимизации. Аналитические методы решения этих задач связаны с решением уравнений, вытекающих из необходимого условия экстремума.

Пусть $f(x)$ — дважды непрерывно дифференцируемая функция переменных $x=(x_1, x_2, ..., x_n)$ в открытой области D. Тогда, если в точке $x_* \in D$ функция $f(x)$ достигает минимума или максимума, в этой точке

$$\frac{\partial f}{\partial x_1}(x_1, ..., x_n)=0, ..., \frac{\partial f}{\partial x_n}(x_1, ..., x_n)=0. \qquad (5.3.24)$$

Соотношения (5.3.24) дают уравнения для определения искомых точек экстремума. Затем следует проверить знак второго дифференциала d^2f в точках эстремума. Если $d^2f>0$, то в этой точке локальный минимум, если $d^2f<0$ — максимум. Если локальных минимумов или максимумов несколько, то из них выбирается точка x_*, которая доставляет наименьшее значение $f(x_*)$; эта точка называется *точкой глобального минимума, аналогично определяется точка x_* глобального максимума.*

Рассмотрим случай, когда $f(x)$ является следующей функцией:

$$f(x)=\sum_{i,j=1}^{n} a_{i,j}x_ix_j+\sum_{i=1}^{n} b_ix_i+c,$$

$a_{i,j}$, b_i, c — вещественные константы, т. е. $f(x)$ — сумма константы c, линейной формы $\sum_{i=1}^{n} b_ix_i$ и квадратичной формы $\sum_{i,j=1}^{n} a_{i,j}x_ix_j$, где $A(a_{i,j})$ — симметричная матрица. В этом случае уравнения (5.3.24) в матричной записи имеют вид

$$Ax+b=0,$$

где $b=(b_1, ..., b_n)$.

Допустим, что систему линейных уравнений можно решить аналитически. Тогда остается проверить знакоопределенность квадратичной формы, поскольку $d^2f>0$ тогда и только тогда, когда квадратичная форма положительно определена.

Для положительной определенности квадратичной формы необходимо и достаточно, чтобы все главные миноры матрицы A были положительны:

$$a_{11}>0, \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}>0, ... \begin{vmatrix} a_{11} & \cdots & a_{1,n} \\ & \cdots & \\ a_{n,1} & \cdots & a_{n,n} \end{vmatrix}>0.$$

5.3.17. Решение обыкновенных дифференциальных уравнений (ОДУ). Задача Коши. Многие задачи естествознания могут быть сформулированы как задача Коши для ОДУ. Определение поведения механической системы во времени с заданными начальными условиями и функцией Лагранжа, а также концентраций реагирующих веществ с заданными уравнениями химической кинетики и начальными концентрациями и многие другие проблемы сводятся

к задаче Коши для ОДУ. Часто роль независимого переменного, обозначаемого нами x, играет время.

Задача Коши для системы ОДУ формулируется следующим образом: требуется найти вектор-функцию $y(x)=(y_1(x), y_2(x), ..., ..., y_n(x))$, непрерывно дифференцируемую на интервале $a \leqslant x \leqslant b$, удовлетворяющую системе уравнений

$$\frac{dy_i}{dx}=f_i(x, y_1, y_2, ..., y_n), \quad 1 \leqslant i \leqslant n, \tag{5.3.25}$$

и начальным условиям

$$y_i(a)=y_i^0. \tag{5.3.26}$$

Заметим, что условия (5.3.26) могут быть заданы в любой точке интервала $[a, b]$, важно, чтобы все компоненты $y(x)$ были заданы в этой точке.

Известны условия, обеспечивающие существование и единственность решения задачи Коши, например такие: если $f(x, y)$ непрерывна в некоторой окрестности D точки (a, y^0) и существуют в D ограниченные производные $\frac{\partial f_i}{\partial y_j}(x, y)$, $1 \leqslant i, j \leqslant n$, то для достаточно малых интервалов $[a, b]$ ($(b-a)$ мало) существует единственное решение задачи Коши.

Условия, приведенные выше, а также им аналогичные обеспечивают лишь локальное существование непрерывно дифференцируемого решения.

Простой пример задачи Коши

$$\frac{dy}{dx}=1+y^2, \quad y(0)=0,$$

имеющей решение $y=\operatorname{tg} x$, показывает, что непрерывно дифференцируемое решение существует лишь на интервале $(0, \pi/2)$, так что локальные условия существования имеют небольшую практическую ценность.

Для линейных систем ОДУ

$$\frac{dy_i}{dx}=\sum_{j=1}^{n} a_{i,j}(x)\, y_j+f_i(x), \quad 1 \leqslant i \leqslant n, \tag{5.3.27}$$

с непрерывными функциями $a_{i,j}(x)$, $f_i(x)$ на интервале $[a, b]$ задача (5.3.27), (5.3.26) имеет единственное решение для любого интервала $[a, b]$ (глобальное существование решения). В этом факте заключается глубокое различие между линейными и нелинейными ОДУ общего вида.

Существует большое число аналитических приемов решения частных ОДУ, которые можно найти в [11]. Рассмотрим только методы, применимые для широких классов ОДУ, а именно:
1) систем линейных уравнений с постоянными коэффициентами;
2) систем линейных уравнений с переменными коэффициентами;
3) систем нелинейных уравнений.

1) Рассмотрим задачу Коши

$$\frac{dy_i}{dz}=\sum_{j=1}^{n} a_{i,j}y_j+f_i(x),\quad 1\leqslant i\leqslant n,$$
$$y_i(a)=y^0,\quad a\leqslant x\leqslant b, \tag{5.3.28}$$

где вещественная матрица $A=(a_{i,j})$ постоянна. Аналитическое решение этой задачи можно записать по аналогии со скалярным уравнением в виде

$$y(x)=e^{A(x-a)}y^0+\int_a^x e^{A(x-s)}f(s)\,ds, \tag{5.3.29}$$

где e^{Ax} — матричная экспонента (см. п. 5.3.10), $y^0=y_1^0, y_2^0, \ldots, y_n^0$, $f(s)=(f_1(s), \ldots, f_n(s))$.

Однако аналитическое представление e^{Ax} через элементы матрицы A в конечном виде возможно лишь для матриц простой структуры, в частности, диагональных. Поэтому систему (5.3.28) предварительно приводят к системе с матрицей простой структуры.

Предположив, что все собственные значения матрицы A различны, обозначим вещественные собственные значения λ_i, $1\leqslant i\leqslant k$; комплексные $\gamma_j^{\pm}=\alpha_j\pm i\beta_j$, $1\leqslant j\leqslant(n-k)/2$. Тогда существует невырожденная вещественная матрица T такая, что замена

$$y=Tz$$

приводит (5.3.28) к системе уравнений

$$\frac{dz}{dx}=T^{-1}ATz+T^{-1}f=Bz+T^{-1}f \tag{5.3.30}$$

с матрицей B простой структуры вида

$$B=\begin{pmatrix}
\lambda_1 & & & & & & 0\\
& \lambda_2 & & & & & \\
& & \ddots & & & & \\
& & & \lambda_k & & & \\
& & & & \alpha_1\beta_1 & & \\
& & & & -\beta_1\alpha_1 & \ddots & \\
& & & & & & \alpha_{n-k/2}\beta_{n-k/2}\\
0 & & & & & & -\beta_{n-k/2}\alpha_{n-k/2}
\end{pmatrix},$$

для которой матричная экспонента легко определяется:

$$e^{Bx}=\begin{pmatrix}
e^{\lambda_1 x} & & & & & \ldots & 0\\
& \ddots & & & & & \\
0\ldots & e^{\lambda_k x} & & & & & 0\\
0\ldots & & e^{\alpha_1 x}\begin{pmatrix}-\cos\beta_1 x\sin\beta_1 x\\ -\sin\beta_1 x\cos\beta_1 x\end{pmatrix}\ldots & & & & 0\\
& & & \ddots & & & \\
0 & \ldots & & & e^{\alpha_{n-k/2}x}\begin{pmatrix}\cos\beta_{n-k/2}x\sin\beta_{n-k/2}x\\ -\sin\beta_{n-k/2}x\cos\beta_{n-k/2}x\end{pmatrix}
\end{pmatrix}.$$

Решение (5.3.30) с начальным условием

$$z(a)=T^{-1}y^0$$

записывается аналогично (5.3.29):

$$z(x)=e^{B(x-a)}T^{-1}y^0+\int_a^x e^{B(x-s)}T^{-1}f(s)\,ds$$

с известной матричной экспонентой e^{Bx}, а отсюда искомое решение $y(x)$ находится умножением на T:

$$y(x)=Te^{B(x-a)}T^{-1}y^0+\int_a^x Te^{B(x-s)}T^{-1}f(s)\,ds. \qquad (5.3.31)$$

Если неоднородность $f(x)$ такова, что интегралы в (5.3.31) берутся аналитически, и матрица T известна, то формула (5.3.31) дает аналитическое решение задачи Коши (5.3.28). Для применения формулы (5.3.31) должна быть решена задача определения собственных значений матрицы A, которая, как известно, для больших n аналитически практически неразрешима.

2) Задача Коши для системы линейных уравнений с переменными коэффициентами (5.3.27), (5.3.26) может быть сведена к (5.3.28) следующим образом.

Запишем (5.3.27) эквивалентным образом:

$$\frac{dy}{dx}=A_0y+f(x)+(A(x)-A_0)\,y,$$

где A_0 — постоянная матрица с различными собственными значениями такая, что

$$\max_{a\leqslant x\leqslant b}\|A(x)-A_0\|\leqslant K,$$

и для матрицы A_0 известно преобразование T к матрице простой структуры B с известной матричной экспонентой. Тогда, согласно (5.3.31), исходная задача Коши эквивалентна интегральному уравнению

$$y(x)=Te^{B(x-a)}T^{-1}y^0+\int_a^x Te^{B(x-s)}T^{-1}\left(f(s)+(A(s)-A_0)\,y(s)\right)ds,$$

которое можно решать методом последовательных приближений:

$$y_{k+1}(x)=Te^{B(x-a)}T^{-1}y_0+\int_a^x Te^{B(x-s)}T^{-1}\left(f(s)+\right.$$
$$\left.+A(s)-A_0)\,y_k(s)\right)ds, \qquad (5.3.32)$$
$$k=0,\ 1,\ 2,\ \ldots;\quad y_0(x)\equiv 0.$$

Можно показать, что для любого интервала $[a, b]$ последовательность $\{y_k(x)\}$ сходится в пространстве функций $C[a, b]$ с нормой

$$\|y\|=\max_{a\leqslant x\leqslant b}\|y(x)\|=\max_{a\leqslant x\leqslant b}\ \max_{1\leqslant i\leqslant n}\left(|y_i(x)|\right)$$

к точному решению задачи Коши $y(x)$

$$y(x)=\lim_{k\to\infty} y_k(x).$$

Скорость сходимости определяется константой K; она тем выше, чем меньше K.

Задача Коши для линейных систем ОДУ может быть записана в виде интегрального уравнения

$$y(x)=y^0+\int_a^x (A(s)y(s)+f(s))\,ds,$$

которое также можно решать методом последовательных приближений

$$y_{k+1}(x)=y^0+\int_a^x (A(s)y_k(s)+f(s))\,ds, \qquad (5.3.33)$$

$$k=0,\ 1,\ 2,\ \ldots;\quad y_0(x)\equiv y^0.$$

Теорема 5.5. *Последовательность* $\{y_k(x)\}$, *определяемая* (5.3.33), *сходится к точному решению* $y(x)$ *задачи Коши* (5.3.27) (5.3.26) *в пространстве* $C[a, b]$:

$$y(x)=\lim_{k\to\infty} y_k(x).$$

Доказательство. Покажем существование предела $\{y_k(x)\}$. Для этого докажем сходимость ряда

$$y_0+(y_1-y_0)+(y_2-y_1)+\ldots+(y_k-y_{k-1})+\ldots \qquad (5.3.34)$$

Так как k-я частичная сумма ряда $s_k=y_k$, то отсюда последует существование предела $\{y_k(x)\}$. Оценим нормы членов ряда. Имеем

$$\|y_1-y_0\|=\|\int_a^x (A(s)y^0+f(s))\,ds\|\leqslant M(x-a),$$

где константой M обозначена оценка

$$\|A(x)y^0+f(x)\|\leqslant\|A(x)\|\,\|y^0\|+\|f(x)\|\leqslant M.$$

Пусть $\|A(x)\|\leqslant M_1$. Тогда

$$\|y_2-y_1\|=\|\int_a^x A(s)(y_1(s)-y_0(s))\,ds\|\leqslant\int_a^x\|A(s)\|\,\|y_1(s)-y_0(s)\|\,ds\leqslant$$

$$\leqslant\int_a^x M_1M(s-a)\,ds=\frac{M_1M}{1\cdot 2}(x-a)^2.$$

Аналогично,

$$\|y_3-y_2\|\leqslant\int_a^x\|A(s)\|\,\|y_2(s)-y_1(s)\|\,ds\leqslant\frac{M_1^2M}{1\cdot 2\cdot 3}(x-a)^3.$$

Отсюда по индукции можно показать, что

$$\|y_k-y_{k-1}\|\leqslant\frac{MM_1^{k-1}}{k!}(x-a)^k.$$

Из (5.3.34) следует, что каждый член ряда начиная со второго по норме $C[a, b]$ меньше соответствующего члена числового ряда с положительными членами

$$M(b-a)+M\frac{M_1(b-a)^2}{2!}+M\frac{M_1^2(b-a)^3}{3!}+\ldots+M\frac{M_1^{k-1}(b-a)^k}{k!}+\ldots,$$

который сходится к сумме

$$S=Me^{M_1(b-a)}.$$

Следовательно, в силу критерия Вейерштрасса ряд (5.3.34) сходится равномерно для всех $x\in[a, b]$. Каждый член ряда (5.3.34) — непрерывная функция от x, так как интеграл есть непрерывная функция верхнего предела, а поэтому предел $\{y_k(x)\}$ существует и является непрерывной функцией x. Покажем, что предел $y(x)=\lim\limits_{k\to\infty} y_k(x)$ удовлетворяет (5.3.27) (начальное условие (5.3.26) для $y(x)$, очевидно, выполнено). Для этого в (5.3.23) перейдем к пределу в левой и правой частях равенства при $k\to\infty$. Причем из равномерной сходимости $\{y_k\}$ следует возможность перейти к пределу под знаком интеграла и получить

$$y(x)=y^0+\int_a^x (A(s)y(s)+f(s))\,ds.$$

Таким образом, $y(x)$ — решение задачи Коши, что и требовалось доказать.

Заметим, что доказана не только сходимость последовательности $\{y_k(x)\}$, но и оценена скорость сходимости; $\{y_k(x)\}$ сходится к пределу не медленнее ряда для экспоненты в точке $M_1(b-a)$.

В том случае, когда $M_1(b-a)$ велико, применяется более сложный для вычислений метод (5.3.32), который при определенных условиях может дать более высокую скорость сходимости.

3) Метод последовательных приближений для задачи (5.3.25), (5.3.26), аналогичный (5.3.33), определяется формулой

$$y_{i,k}(x)=y_i^0+\int_a^x f_i(s, y_{1,k}(s), y_{2,k}(s), \ldots, y_{n,k}(s))\,ds, \qquad (5.3.35)$$

$$k=0, 1, 2, \ldots;\ y_{i,k}(x)\equiv y_i^0,\ 1\leqslant i\leqslant n.$$

Если все интегралы в (5.3.35) для $k=0, 1, 2, \ldots$ вычисляются аналитически, то решение $y_i(x)$ находится в аналитическом виде как предел

$$y_i(x)=\lim_{k\to\infty} y_{i,k}(x).$$

Аналогично доказательству теоремы 5.5 можно доказать существование предела при условиях, обеспечивающих существование и единственность решения задачи Коши.

Для примера решим задачу

$$\frac{dy}{dx}=1+y^2,\quad y(0)=0.$$

Имеем

$$y_{k+1}(x)=\int_0^x\left(1+y_k^2(s)\right)ds,\quad y_0(s)\equiv 0,$$

$$y_1(x)=x,\quad y_2(x)=\int_0^x(1+s)^2\,ds=x+\frac{x^3}{3},$$

$$y_3(x)=\int_0^x\left(1+\left(s+\frac{s^3}{3}\right)^2\right)ds=x+\frac{x^3}{3}+\frac{2}{15}x^5+\frac{x^7}{63},\ \ldots\ .$$

Таким образом, получаем правую часть следующего тождества (при $k\to\infty$):

$$\operatorname{tg}x=x+\frac{x^3}{3}+\frac{2}{15}x^5+\ldots+\frac{2^{2k}\left(2^{2k}-1\right)B_k}{(2k)!}x^{2k-1}+\ldots\ ,$$

где B_k — числа Бернулли; ряд сходится для $x<\pi/2$, т. е. при $k\to\infty$ получаем точное решение задачи.

Вторым аналитическим методом решения нелинейной задачи Коши является метод разложения решения в ряд Тейлора. Если метод последовательных приближений основан на процедуре интегрирования, то метод рядов Тейлора основан на дифференцировании.

Пусть $f(x, y)$ имеет m непрерывных производных по всем аргументам в области D. Решение $y_i(x)$ ищется в виде отрезка ряда Тейлора:

$$y_{im}(x)=y_i(a)+\frac{dy_i}{dx}(a)(x-a)+\frac{1}{2!}\frac{d^2y_i}{dx^2}(a)(x-a)^2+\ldots+$$

$$+\frac{1}{m!}\frac{d^my_i}{dx^m}(a)(x-a)^m. \qquad (5.3.36)$$

Из исходной задачи получаем

$$y_i(a)=y_i^0,$$

$$\frac{dy_i}{dx}(a)=f_i(a, y_1^0, \ldots, y_n^0).$$

Затем из (5.3.25) находим

$$\frac{d^2y_i}{dx^2}=\frac{\partial f_i}{\partial x}+\sum_{j=1}^{n}\left(\frac{\partial f_i}{\partial y_j}\right)f_j$$

с последующей подстановкой $x=a$, y_i^0, затем $\dfrac{d^3y_i(a)}{dx^3}$ и т. д. Остаточный член ряда Тейлора дает погрешность приближения $y_{i,m}(x)$ к точному решению $y_i(x)$ задачи Коши.

Для примера решим методом рядов Тейлора задачу

$$\frac{dy}{dx}=x+e^{\alpha y},\ y(0)=0,$$

где α — параметр. С точностью до $O(x^4)$ получаем

$$y_3(x)=x+(1+\alpha)\frac{x^2}{2}+\frac{\alpha(1-2\alpha)}{6}x^3+O(x^4).$$

Обратим внимание на то, что аналитическое решение дает приближенную формулу для произвольных значений параметра α; такой результат нельзя получить в рамках численных методов.

5.3.18. Решение ОДУ. Краевые задачи. Краевые задачи отличаются от задачи Коши тем, что дополнительные условия для дифференциальных уравнений задаются не в одной точке, а в двух: на краях интервала $a\leqslant x\leqslant b$, т. е. в точках $x=a$, $x=b$.

Покажем, как в приложениях возникают краевые задачи.

Пример 1. Найти траекторию механической системы, связывающую начальное и конечное состояния системы, задаваемые обобщенными координатами q_i, $1\leqslant i\leqslant n$, в момент времени $t=t_0$: $q_i(t_0)=q_i^0$, в момент времени $t=t_1$: $q_i(t_1)=q_i'$. Эта задача эквивалентна решению уравнений Лагранжа

$$\frac{d}{dt}\left(\frac{\partial L}{\partial q_i'}\right)-\frac{\partial L}{\partial q_i}=0,\ t_0<t<t_1,$$

с дополнительными условиями

$$q_i(t_0)=q_i^0,\ q_i(t_1)=q_i',\ 1\leqslant i\leqslant n.$$

Роль независимой переменной x здесь играет время t, на концах интервала $[t_0,\ t_1]$ задаются дополнительные условия для определения решения системы ОДУ — уравнений Лагранжа.

Пример 2. Найти химическую кинетику реакции с заданной концентрацией некоторых исходных и конечных продуктов реакции (кинетика реакции — это решение системы ОДУ $c_i(t)$, удовлетворяющее краевым условиям).

Уравнения химической кинетики записываются в виде

$$\frac{dc_i}{dt}=f_i(c_1,\ \ldots,\ c_n),\ t_0<t<t_1,\ 1\leqslant i\leqslant n,$$

с начальными условиями для концентраций c_i

$$c_i(t_0)=c_i^0,\ 1\leqslant i\leqslant n-k,\ 1\leqslant k\leqslant n-1,$$

и конечными условиями для концентраций c_i

$$c_i(t_1)=c_i^1,\ n-k+1\leqslant i\leqslant n.$$

Число дополнительных условий на левом и правом концах интервала $[t_0,\ t_1]$ равно порядку n системы ОДУ.

Другим важным источником краевых задач является метод разделения переменных решения уравнений с частными производными

(п. 5.3.19). Во многих случаях метод приводит к решению линейного дифференциального уравнения вида

$$a(x)\frac{d^2y}{dx^2}+b(x)\frac{dy}{dx}+c(x)y+\lambda q(x)y=f(x) \qquad (5.3.37)$$

на интервале $a<x<b$ (λ — параметр) со следующими условиями на краях интервала:

$$\alpha_0 y(a)+\alpha_1\frac{dy}{dx}(a)=\alpha_2;$$

$$\beta_0 y(b)+\beta_1\frac{dy}{dx}(b)=\beta_2, \qquad (5.3.38)$$

где α_i, β_i — заданные константы, $\alpha_0^2+\alpha_1^2\neq 0$, $\beta_0^2+\beta_1^2\neq 0$.

Если однородное уравнение ($f(x)\equiv 0$), соответствующее (5.3.37), имеет коэффициентами $a(x)$, $b(x)$, $c(x)$, $g(x)$ — полиномы, то для некоторых из них общее решение выражается через специальные функции (см. [11]). В таких случаях аналитическое решение краевой задачи (5.3.37), (5.3.38) сводится к вычислению интегралов от произведения $f(x)$ на комбинации специальных функций и решению системы линейных алгебраических уравнений второго порядка.

Пусть, например, в (5.3.37) $a(x)=g(x)\equiv 1$, $b(x)=c(x)=0$, λ — вещественный параметр, $\lambda<0$. Обозначим $\lambda=-\nu^2$ и положим в (5.3.38) $\alpha_i=\beta_i=1$, $i=1,2$, $a=0$, $b=1$. Имеем краевую задачу

$$\frac{d^2y}{dx^2}-\nu^2 y=f(x),$$

$$y(0)+\frac{dy}{dx}(0)=1;\quad y(1)+\frac{dy}{dx}(1)=1.$$

Решение $y(x)$ ищем в виде суммы

$$y(x)=u(x)+w(x),$$

где $u(x)$ является решением краевой задачи

$$\frac{d^2u}{dx^2}-\nu^2 u=0,$$

$$u(0)+\frac{du}{dx}(0)=1,\quad u(1)+\frac{du}{dx}(1)=1-w(1)-\frac{dw}{dx}(1)$$

с однородным уравнением и неоднородными краевыми условиями, $w(x)$ — решением задачи Коши

$$\frac{d^2w}{dx^2}-\nu^2 w=f(x),\quad w(0)=0,\quad \frac{dw}{dx}(0)=0.$$

Решим последнюю задачу методом вариации произвольных постоянных. Имеем

$$w(x)=c_1(x)e^{\nu x}+c_2(x)e^{-\nu x}.$$

Тогда

$$\frac{dw}{dx}(x)=\frac{dc_1}{dx}e^{\nu x}+\frac{dc_2}{dx}e^{-\nu x}+c_1\nu e^{\nu x}-c_2\nu e^{-\nu x}.$$

Положим

$$\frac{dc_1}{dx}e^{\nu x}+\frac{dc_2}{dx}e^{-\nu x}=0; \tag{5.3.39}$$

следовательно,

$$\frac{d^2w}{dx^2}=\frac{dc_1}{dx}\nu e^{\nu x}-\frac{dc_2}{dx}\nu e^{-\nu x}+c_1\nu^2e^{\nu x}+c_2\nu^2e^{-\nu x}.$$

Отсюда и из исходного уравнения для w находим

$$\frac{dc_1}{dx}\nu e^{\nu x}-\frac{dc_2}{dx}\nu e^{-\nu x}=f(x). \tag{5.3.40}$$

Из (5.3.39), (5.3.40) определяем

$$\frac{dc_1}{dx}=-e^{-2\nu x}\frac{dc_2}{dx},$$

$$\left(-e^{-2\nu x}\nu e^{\nu x}-\nu e^{-\nu x}\right)\frac{dc_2}{dx}=f(x),$$

откуда

$$\frac{dc_2}{dx}=-\frac{e^{\nu x}}{2\nu}f(x),\quad c_2(x)=c_2^0-\int_0^x\frac{1}{2\nu}e^{\nu s}f(s)\,ds,$$

$$\frac{dc_1}{dx}=\frac{e^{-\nu x}}{2\nu}f(x),\quad c_1(x)=c_1^0+\int_0^x\frac{1}{2\nu}e^{-\nu s}f(s)\,ds.$$

Значения констант c_1^0, c_2^0 находим из начальных условий для $w(x)$:

$$0=w(0)=\left[c_1^0-\int_0^x\frac{1}{2\nu}e^{-\nu s}f(s)\,ds\right]e^{\nu x}+\left[c_2^0-\int_0^x\frac{1}{2\nu}e^{\nu s}f(s)\,ds\right]e^{-\nu x}=c_1^0+c_2^0,$$

$$0=\frac{dw}{dx}(0)=\left[c_1^0-\int_0^x\frac{1}{2\nu}e^{-\nu s}f(s)\right]\nu e^{\nu x}-\left[c_2^0-\int_0^x\frac{1}{2\nu}e^{\nu s}f(s)\,ds\right]\nu e^{-\nu x}=c_1^0\nu-c_2^0\nu.$$

Получаем $c_1^0=0$, $c_2^0=0$. Окончательно имеем

$$w(x)=+e^{\nu x}\int_0^x\frac{1}{2\nu}e^{-\nu s}f(s)\,ds-e^{-\nu x}\int_0^x\frac{1}{2\nu}e^{\nu s}f(s)\,ds,$$

$$\frac{dw}{dx}(x)=+e^{\nu x}\int_0^x\frac{1}{2}e^{-\nu s}f(s)\,ds+e^{-\nu x}\int_0^x\frac{1}{2}e^{\nu s}f(s)\,ds.$$

Для функции $u(x)$ краевое условие в точке $x=1$ перепишем с учетом формул для $w(x)$, $\frac{dw}{dx}$ так:

$$u(1)+\frac{du}{dx}(1)=1+\int_0^1\left[-\frac{1+2\nu}{2\nu}e^{-\nu(s-1)}-\frac{1-2\nu}{2\nu}e^{\nu(s-1)}\right]f(s)\,ds.$$

Решение краевой задачи

$$u(x)=c_1e^{\nu x}+c_2e^{-\nu x}$$

сводится к определению констант c_1, c_2 из краевых условий

$$c_1+c_2=1,$$

$$c_1\nu e^{\nu}-c_2\nu e^{-\nu}=1+\int_0^1\frac{1}{2\nu}\left[-(1+2\nu)e^{-\nu(s-1)}-\right.$$

$$\left.-(1-2\nu)e^{\nu(s-1)}\right]f(s)ds. \qquad (5.3.41)$$

Отсюда находим c_1, c_2 и искомое решение $y(x)$:

$$y(x)=\frac{e^{\nu x}}{-\nu(e^{-\nu}+e^{\nu})}\left\{-\nu e^{-\nu}-1-\int_0^1\frac{1}{2\nu}\left[-(1+2\nu)e^{-\nu(s-1)}-(1-2\nu)e^{\nu(s-1)}\right]\times\right.$$

$$\left.\times f(s)\,ds\right\}+\frac{e^{-\nu x}}{-\nu(e^{-\nu}+e^{\nu})}\left\{-\nu e^{\nu}+1+\int_0^1\frac{1}{2\nu}\left[-(1+2\nu)e^{-\nu(s-1)}-\right.\right.$$

$$\left.\left.-(1-2\nu)e^{\nu(s-1)}\right]f(s)\,ds\right\}-e^{\nu x}\int_0^x\frac{1}{2\nu}e^{-\nu s}f(s)\,ds+e^{-\nu x}\int_0^x\frac{1}{2\nu}e^{\nu s}f(s)\,ds.$$

Рассмотренная краевая задача имеет единственное решение. Этот факт связан с тем, что определитель системы линейных уравнений (5.3.41) отличен от нуля:

$$-\nu(e^{-\nu}+e^{\nu})\neq 0,\quad \nu\neq 0.$$

Можно привести примеры неразрешимых краевых задач, а также краевых задач, имеющих бесчисленное множество решений. Эти свойства задач определяются системой линейных алгебраических уравнений типа (5.3.41), так как неразрешимость или бесчисленное множество решений системы приводит к аналогичному свойству исходной краевой задачи.

В том случае, когда решение однородного уравнения, соответствующего (5.3.37), не выражается в известных элементарных и специальных функциях, можно применять эффективный метод решения (5.3.37), (5.3.38) — *метод прогонки*. Суть этого метода состоит в сведении краевой задачи к некоторым задачам Коши.

Для простоты изложения рассмотрим уравнение

$$\frac{d^2y}{dx^2}+b(x)\frac{dy}{dx}+c(x)y=f(x) \tag{5.3.42}$$

с краевыми условиями

$$\frac{dy}{dx}(a)=\alpha_0 y(a)+\alpha_2; \tag{5.3.43}$$

$$\frac{dy}{dx}(b)=\beta_0 y(b)+\beta_2. \tag{5.3.44}$$

Будем искать линейную комбинацию

$$\frac{dy}{dx}=A(x)y+B(x) \tag{5.3.45}$$

(с неизвестными пока функциями $A(x)$, $B(x)$) такую, чтобы при подстановке (5.3.45) в уравнение (5.3.42) оно стало тождеством. Для этого в соотношении

$$\frac{dA}{dx}y+A(Ay+B)+\frac{dB}{dx}+B(Ay+B)+cy=f$$

приравняем отдельно коэффициенты при y^1, y^0. Получим уравнения для $A(x)$, $B(x)$ $(a<x<b)$ с начальными условиями из (5.3.43):

$$\frac{dA}{dx}=-A^2-bA-c,\quad A(a)=\alpha_0; \tag{5.3.46}$$

$$\frac{dB}{dx}=-AB-bB+f,\quad B(a)=\alpha_2. \tag{5.3.47}$$

Предположим, что решение задачи Коши (5.3.46) $A(x)$ существует на интервале $a\leqslant x\leqslant b$ и может быть определено аналитически. Тогда решение (5.3.47) записывается явно

$$B(x)=\alpha_2 e^{-\int_a^x (A(s)+b(s))\,ds}+\int_a^x e^{-\int_s^x (A(s)+b(s))\,ds} f(s)\,ds.$$

Предположим, что $A(b)-\beta_0\neq 0$. Тогда из (5.3.44), (5.3.45) находим

$$y(b)=\frac{\beta_2-B(b)}{A(b)-\beta_0}. \tag{5.3.48}$$

На этом завершается прямая прогонка, смысл которой состоит в перегоне условия (5.3.43) слева направо от точки $x=a$ в точку $x=b$ и в определении $y(b)$.

Обратная прогонка — это решение второй задачи Коши: уравнения (5.3.45) с условиями (5.3.48) на правом конце интервала $a\leqslant x\leqslant b$. Окончательно получаем решение исходной краевой задачи

$$y(x)=y(b)e^{\int_b^x A(s)\,ds}+\int_b^x e^{\int_s^x A(s)\,ds} B(s)\,ds,\quad a\leqslant x\leqslant b.$$

В том случае, когда принятое предположение не выполняется, краевая задача либо неразрешима, либо имеет бесконечно много решений.

Из аналитических методов решения нелинейных краевых задач рассмотрим *метод сведения к интегральному уравнению*. Пусть имеем краевую задачу

$$\frac{d^2y}{dx^2}=f(x, y),\ \ y(0)=y(b)=0. \tag{5.3.49}$$

Если бы правая часть была известной функцией от x, например $F(x)$,

$$\frac{d^2y}{dx^2}=F(x),\ \ y(0)=y(b)=0,$$

то с помощью подстановки проверяется, что решение краевой задачи имеет вид

$$y(x)=\int_0^b K(x, s)F(s)\,ds,$$

где

$$K(x, s)=\begin{cases}\dfrac{x(s-b)}{b}, & x\leqslant s\leqslant b,\\ \dfrac{s(x-b)}{b}, & 0\leqslant s\leqslant x.\end{cases}$$

Используя эту формулу, сводим задачу (5.3.49) к решению следующего интегрального уравнения:

$$y(x)=\int_0^b K(x, s)f(y(s), s)\,ds.$$

Если метод последовательных приближений

$$y_{k+1}(x)=\int_0^b K(x, s)f(y_k(s), s)\,ds$$

($k=0,\ 1,\ 2,\ \ldots$; $y_0(x)$—заданная функция) сходится к точному решению $y(x)$ и для любого k интеграл в правой части можно вычислить аналитически, то точное решение

$$y(x)=\lim_{k\to\infty} y_k(x).$$

Вопрос о единственности требует специального рассмотрения. Обычно он снимается в результате анализа прикладной задачи.

Например, для краевой задачи

$$\frac{d^2y}{dx^2}=-y^2+yx^2+2-x^2(x-1),\ \ y(0)=y(1)=0$$

метод последовательных приближений

$$y_{k+1}(x)=(x-1)\int_0^x s[-y_k^2+y_k s^2+2-s^2(s-1)]\,ds+\int_x^1 (s-1)[-y_k^2+y_k s^2+$$
$$+2-s^2(s-1)]\,ds,\quad k=0,\ 1,\ 2,\ \ldots,\ y_0(x)\equiv 0,$$

дает первое приближение

$$y_1(x)=x^5\left(\frac{1}{5}-\frac{1}{4}\right)+x^4\left(\frac{1}{3}-\frac{1}{4}\right)+x\left(\frac{-31}{30}\right)+x^2,$$

отличающееся от точного решения

$$y(x)=x(x-1)$$

не более чем на величину

$$\max_{0\leqslant x\leqslant 1}|y(x)-y_1(x)|=\max_{0\leqslant x\leqslant 1}\left|-\frac{x}{30}+\frac{x^4}{12}-\frac{x^5}{20}\right|\sim 0{,}02.$$

Заметим, что эта точность достигнута в результате небольшого объема вычислений, а приближение найдено в виде формулы. Для сравнения «вычислительных затрат» можно рекомендовать решить эту задачу разностным методом (см. 10.4) с той же точностью.

5.3.19. Решение уравнений с частными производными. Из аналитических методов решения задач для уравнений с частными производными рассмотрим *метод Фурье* (*метод разделения переменных*) на примерах основных линейных уравнений математической физики и *метод подобия* для нелинейного уравнения теплопроводности. Для дальнейшего изучения рассматриваемых вопросов рекомендуется [31].

1. Волновое уравнение. Малые поперечные колебания однородной струны описываются уравнением

$$\frac{\partial^2 u}{\partial t^2}=a^2\frac{\partial^2 u}{\partial x^2}+f(x,\ t),\qquad (5.3.50)$$

где независимые переменные $(x,\ t)\in D=\{0\leqslant x\leqslant l,\ 0\leqslant t\leqslant t_1\}$, решение $u(x,\ t)$ — это отклонение от положения равновесия $(u=0)$ струны в точке x в момент времени t (рис. 5.4). Здесь константа a^2 определяется формулой

$$a^2=T/\rho,$$

где T — сила натяжения струны, ρ — ее плотность.

Неоднородность $f(x,\ t)=F(x,\ t)/\rho$; $F(x,\ t)$ — нагрузка, действующая на струну в точке x в момент времени t.

Для определения решения $u(x,\ t)$ задаются начальные условия

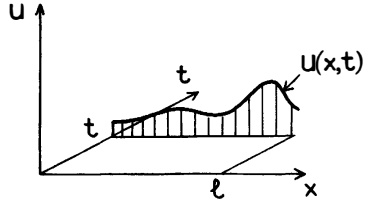

Рис. 5.4

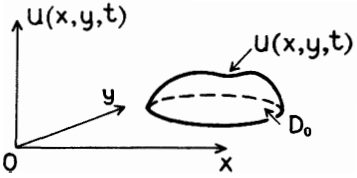

Рис. 5.5

$$u(x, 0)=\varphi_0(x), \quad 0\leqslant x\leqslant l,$$
$$\frac{\partial u}{\partial t}(x, 0)=\varphi_1(x). \tag{5.3.51}$$

Здесь $\varphi_0(x)$ — начальная форма струны, $\varphi_1(x)$ — начальная скорость. Кроме того, следует задать граничные условия. Например, если левый конец струны закреплен, а правый свободен, то граничные условия следующие:

$$u(0,\ t)=0,\ \frac{\partial u}{\partial x}(l,\ t)=0;$$

если закреплены два конца, то

$$u(0,\ t)=0,\ u(l,\ t)=0. \tag{5.3.52}$$

Уравнение (5.3.50) — одномерное волновое уравнение, так как в нем одна пространственная переменная x.

Малые поперечные колебания мембраны описывает двумерное волновое уравнение

$$\frac{\partial^2 y}{\partial t^2}=a^2\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 y}{\partial y^2}\right)+f(x,\ y,\ t).$$

Здесь $(x,\ y)\in D_0$, $0\leqslant t\leqslant t_1$, $u(x,\ y,\ t)$ — отклонение мембраны от нуля (положение равновесия мембраны) в точке x, y в момент времени t (рис. 5.5).

Распространение волн в трехмерной однородной среде $(a^2=$ $=\text{const})$ описывается трехмерным волновым уравнением

$$\frac{\partial^2 u}{\partial t^2}=a^2\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right)+f(x,\ y,\ z,\ t).$$

2. Уравнение теплопроводности или диффузии. Процесс распространения тепла или диффузии частиц в одномерном случае описывается уравнением

$$\frac{\partial u}{\partial t}=a^2\frac{\partial^2 u}{\partial x^2}+f(x,\ t), \tag{5.3.53}$$

где $(x,\ t)\in D=\{0\leqslant x\leqslant l,\ 0\leqslant t\leqslant t_1\}$. Решение $u(x,\ t)$ — это температура бесконечно тонкого стержня в точке x в момент времени t (для задачи теплопроводности). Коэффициент

$$a^2=k/(c\rho),$$

где k — коэффициент теплопроводности, c — удельная теплоемкость, ρ — плотность материала стержня. Функция

$$f(x,\ t)=\frac{F(x,\ t)}{c\rho},$$

где $F(x,\ t)$ — плотность тепловых источников в точке x в момент времени t (рис. 5.6).

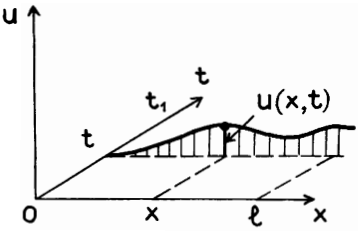

Рис. 5.6

Для однозначного определения решения задаются начальное условие

$$u(x,\ 0)=\varphi_0(x),\ 0\leqslant x\leqslant l \qquad (5.3.54)$$

($\varphi_0(x)$ — начальное распределение температуры в стержне) и граничные условия, например вида

$$u(0,\ t)=\mu_0(t),\ u(l,\ t)=\mu_1(t) \qquad (5.3.55)$$

(μ_0, μ_1 — заданные температурные режимы на краях стержня) или

$$\frac{\partial u}{\partial x}(0,\ t)=\nu_0(t)$$

(задан тепловой поток на левом конце стержня); возможны различные комбинации краевых условий.

Двумерное уравнение теплопроводности или диффузии имеет вид

$$\frac{\partial u}{\partial t}=a^2\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)+f(x,\ y,\ t).$$

Здесь $(x,\ y)\in D$ — плоская область, $0\leqslant t\leqslant t_1$.

Трехмерное уравнение

$$\frac{\partial u}{\partial t}=a^2\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right)+f(x,\ y,\ z,\ t).$$

Здесь $(x,\ y,\ z)\in D$, D_0 — трехмерная область, где ставится задача определения температуры $u(x,\ y,\ z,\ t)$ в момент времени t. Для выделения однозначного решения задаются дополнительные условия, аналогичные (5.3.54), (5.3.55).

Волновое уравнение и уравнение теплопроводности — примеры нестационарных уравнений, описывающих нестационарные процессы (есть зависимость решений от времени t).

3. Стационарные уравнения. Стационарные процессы описывают решения уравнений с частными производными $u=u(x,\ y,\ z)$, в которых отсутствует зависимость от времени t. Если в волновом уравнении и уравнении теплопроводности положить

$$u(x,\ y,\ z,\ t)=u(x,\ y,\ t),\ f(x,\ y,\ z,\ t)=f(x,\ y,\ z)$$

и считать, что граничные условия не зависят от t, то приходим к стационарному уравнению вида

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}=f(x,\ y,\ z)$$

или

$$\Delta u=f, \qquad (5.3.56)$$

где $\Delta=\dfrac{\partial^2}{\partial x^2}+\dfrac{\partial^2}{\partial y^2}+\dfrac{\partial^2}{\partial z^2}$ — оператор Лапласа.

В отличие от нестационарных уравнений начальные условия для (5.3.56) не задаются, остаются лишь граничные.

Уравнение (5.3.56) называется *уравнением Пуассона.* Однородное $(f \equiv 0)$ уравнение Пуассона

$$\Delta u = 0 \tag{5.3.57}$$

является уравнением Лапласа.

Решение (5.3.56) или (5.3.57) ищется в области D. Для однозначного определения решения задаются на границе Г области граничные условия

$$u|_{\Gamma} = \varphi_0(x,\ y,\ z)$$

(задача Дирихле) либо

$$\left.\frac{\partial u}{\partial n}\right|_{\Gamma} = \varphi_1(x,\ y,\ z)$$

(задача Неймана) или их комбинация.

Перечислим физические процессы, изучение которых сводится к решению уравнений (5.3.56), (5.3.57): распределение стационарного температурного поля в однородной среде, стационарная диффузия, электростатика однородной изотропной среды, магнитостатика, распределение поля постоянного электрического тока, потенциальное течение несжимаемой жидкости [31].

4. Корректность постановки задач для уравнений с частными производными. Задачу называют корректно поставленной, если решение:

1) существует в некотором классе функций;

2) единственное в некотором классе функций;

3) непрерывно зависит от входных данных задачи (начальные, граничные условия, коэффициенты в уравнении и т. д.).

Это определение корректности распространяется и на другие математические задачи. Смысл его состоит в том, что класс корректно поставленных задач составляют только те задачи, решение которых «мало» изменится, если задачу «мало» изменить (возмутить). Математические задачи являются идеализацией естественных явлений, они получаются отбрасыванием многочисленных «не важных» деталей. Поэтому, решая конкретную задачу, необходимо быть уверенным, что она корректно поставлена, а значит, ее решение «близко» к естественному процессу.

Условия, обеспечивающие существование, единственность решения задачи Коши для ОДУ (см. п. 5.3.17) определяют корректно поставленные задачи.

Ниже рассматриваются краевые задачи для уравнений с частными производными также корректно поставленные.

Приведем пример некорректно поставленной задачи для уравнения Лапласа (пример Адамара). Рассмотрим уравнение

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

с условиями

$$u(0,\ y)=0,\ \frac{\partial u}{\partial x}(0,\ y)=\frac{1}{K}\sin Ky.$$

Если перейти в условиях к пределу при $K\to\infty$, то предельная задача имеет решение

$$u(x,\ y)=0.$$

Решение исходной задачи

$$u(x,\ y,\ K)=\frac{1}{K^2}\operatorname{sh} Kx\sin Ky.$$

Заметим, что при $K\to\infty$

$$u(x,\ y,\ K)\nrightarrow 0,$$

а это означает, что нарушено условие непрерывной зависимости решения от входных данных задачи (от K), т. е. задача некорректно поставлена.

Может сложиться впечатление, что решать некорректно поставленные задачи вообще не имеет смысла, а их постановка свидетельствует о неадекватной математической модели изучаемому явлению. Это мнение, широко распространенное до 60-х годов нашего столетия, было опровергнуто в работах советского математика А. Н. Тихонова, который показал естественные источники некорректно поставленных задач и предложил эффективные методы их решения [29, 30].

5. Метод Фурье для уравнения свободных колебаний струны. Рассмотрим задачу (5.3.50), (5.3.51), (5.3.52) с неоднородностью $f\equiv 0$, т. е. свободные колебания закрепленной на концах струны с начальным профилем $\varphi_0(x)$ и начальной скоростью $\varphi_1(x)$.

Ищем частные решения, не равные тождественно нулю, уравнения (5.3.50) в виде

$$u(x,\ t)=X(x)\,T(t).$$

Подставляя в уравнение это выражение, найдем

$$\frac{T''_{t^2}(t)}{a^2T(t)}=\frac{X''_{x^2}(x)}{X(x)}=-\lambda;$$

равенство для функций переменных t и x возможно только тогда, когда λ — постоянное число, откуда получаем два дифференциальных уравнения:

$$T''_{t^2}+a^2\lambda T=0; \tag{5.3.58}$$

$$X''_{x^2}+\lambda X=0. \tag{5.3.59}$$

Чтобы получить не равные тождественно нулю функции $X(x)$ с условиями

$$X(0)=0,\ \ X(l)=0, \tag{5.3.60}$$

вытекающими из (5.3.52), следует решить задачу (5.3.59), (5.3.60) — задачу на собственные значения (задачу Штурма — Лиувилля): определить те значения λ, при которых она имеет нетривиальное решение. Эти значения называются *собственными*, а соответствующие решения $X(x)$ — *собственными функциями*.

Рассмотрев значения $\lambda<0$, $\lambda=0$, $\lambda>0$, можно найти, что собственные значения

$$\lambda_k=\left(\frac{k\pi}{l}\right)^2,\ \ k=1,\ 2,\ \ldots,$$

а соответствующие собственные функции

$$X_k(x)=\sin\frac{k\pi x}{l},\ \ k=1,\ 2,\ \ldots.$$

Отрицательные значения k приводят к собственным функциям, отличающимся множителем (-1), поэтому ниже рассматриваются лишь положительные k.

Для значений $\lambda=\lambda_k$ общее решение уравнения имеет вид (5.3.58):

$$T_k(t)=a_k\cos\frac{k\pi at}{l}+b_k\sin\frac{k\pi at}{l},$$

где a_k, b_k — произвольные константы. Функция

$$u_k(x,\ t)=X_k(x)\,T_k(t)=\left(a_k\cos\frac{k\pi at}{l}+b_k\sin\frac{k\pi at}{l}\right)\sin\frac{k\pi x}{l}$$

удовлетворяет уравнению

$$\frac{\partial^2 u}{\partial t^2}=a^2\frac{\partial^2 u}{\partial x^2}$$

и граничным условиям

$$u(0,\ t)=0,\ \ u(l,\ t)=0$$

при произвольных a_k, b_k. Это же верно и для конечной суммы решений $u_k(x,\ t)$, а также для ряда

$$u(x,\ t)=\sum_{k=1}^{\infty}u_k(x,\ t),$$

если ряд сходится и его можно почленно дважды дифференцировать по x и t.

Определим константы a_k, b_k, чтобы удовлетворить начальным условиям (5.3.51). Для этого следует положить

$$\varphi_0(x)=\sum_{k=1}^{\infty}a_k\sin\frac{k\pi x}{l},\ \ \varphi_1(x)=\sum_{k=1}^{\infty}\frac{k\pi a}{l}b_k\sin\frac{k\pi x}{l}=\left.\frac{\partial u}{\partial t}\right|_{t=0},$$

т. е. найти коэффициенты разложений функций $\varphi_0(x)$, $\varphi_1(x)$ в ряд Фурье; получаем

$$a_k=\frac{2}{l}\int_0^l \varphi_0(x)\sin\frac{k\pi x}{l}dx,\quad b_k=\frac{2}{k\pi a}\int_0^l \varphi_1(x)\sin\frac{k\pi x}{l}dx.$$

Окончательно решение поставленной задачи записывается в виде

$$u(x,\ t)=\sum_{k=1}^{\infty}\left[\left(\frac{2}{l}\int_0^l \varphi_0(x)\sin\frac{k\pi x}{l}dx\right)\cos\frac{k\pi at}{l}+\right.$$
$$\left.+\left(\frac{2}{k\pi a}\int_0^l \varphi_1(x)\sin\frac{k\pi x}{l}dx\right)\sin\frac{k\pi at}{l}\right]\sin\frac{k\pi x}{l}. \qquad (5.3.61)$$

Если начальные функции $\varphi_0(x)$, $\varphi_1(x)$ удовлетворяют условиям

$$\varphi_0(x)\in C^3[0,\ l],\ \varphi_1(x)\in C^2[0,\ l],$$
$$\varphi_0(0)=\varphi_0(l)=0,\ \varphi_0''(0)=\varphi_0''(l)=0,\ \varphi_1(0)=\varphi_1(l)=0,$$

то возможно почленное дифференцирование ряда и полученные ряды сходятся абсолютно и равномерно при $0\leqslant x\leqslant l$ и любом t. Если указанные условия нарушены, то может не существовать дважды непрерывно дифференцируемого решения (задача некорректно поставлена в классе дважды непрерывно дифференцируемых функций; в этом случае решение определяется по-другому [31]).

Предположим, что интегралы в (5.3.61) можно взять аналитически, а ряд просуммировать; тогда получим аналитическое решение поставленной задачи методом Фурье. Очевидно, что если φ_0, φ_1 представляют собой конечные суммы синусоид вида $\sin\frac{m\pi x}{l}$ (m — целое), то ряд (5.3.61) сводится к конечной сумме.

Методом Фурье, по аналогии с рассмотренной задачей, исследуются [31]: вынужденные колебания струны ($f\neq 0$); радиальные колебания газа в сфере и цилиндре; колебания прямоугольной и круглой мембраны и другие задачи. При этом собственные функции задачи Штурма — Лиувилля могут выражаться через специальные функции, например функции Бесселя.

6. Метод Фурье для уравнения теплопроводности. Рассмотрим задачу определения температуры в однородной тонкой пластине, контур которой поддерживается при температуре $u=0$. Начальное распределение температуры задано — $\varphi(x,\ y)$ (рис. 5.7). Эта задача сводится к решению уравнения

$$\frac{\partial u}{\partial t}=a^2\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right) \qquad (5.3.62)$$

в области $D_0=\{0\leqslant x\leqslant p,\ 0\leqslant y\leqslant q\}$, $0\leqslant t\leqslant t_1$, с начальным условием

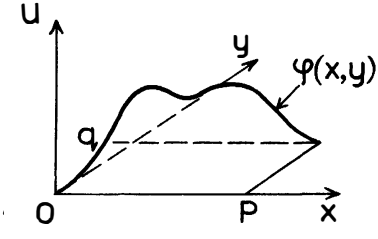

Рис. 5.7

$$u(x,\ y,\ 0)=\varphi(x,\ y),\ \ x,\ \ y\in D_0 \tag{5.3.63}$$

при граничных условиях

$$\begin{aligned} u(x,\ y,\ t)|_{x=0}&=u(x,\ y,\ t)|_{x=p}=0,\ \ 0\leqslant y\leqslant q,\\ u(x,\ y,\ t)|_{y=0}&=u(x,\ y,\ t)|_{y=q}=0,\ \ 0\leqslant x\leqslant p. \end{aligned} \tag{5.3.64}$$

Согласно методу Фурье, частные решения (5.3.62) ищем в виде

$$u=X(x)\,Y(y)\,T(t).$$

Для определения X, Y, T получаем уравнения

$$X''_{x^2}+\lambda^2 x=0,$$

$$Y''_{y^2}+\mu^2 y=0,$$

$$T'_t+a^2(\lambda^2+\mu^2)\,T=0,$$

где λ^2, μ^2 — константы. Общие решения этих уравнений:

$$X(x)=c_1\cos\lambda x+c_2\sin\lambda x,$$

$$Y(y)=c_3\cos\mu x+c_4\sin\mu x,$$

$$T(t)=c_5\,\mathrm{e}^{-a^2(\lambda^2+\mu^2)t}.$$

Граничные условия приводят к следующим собственным значениям:

$$\lambda_m=\frac{m\pi}{p},\ \ \mu_n=\frac{n\pi}{q},\ \ m,\ \ n=1,\ \ 2,\ \ 3\ \ \ldots.$$

Частные решения, удовлетворяющие (5.3.62) и (5.3.64), таковы:

$$u_{m,\,n}=a_{m,\,n}\,\mathrm{e}^{-a^2\pi^2\left(\frac{m^2}{p^2}+\frac{n^2}{q^2}\right)t}\sin\frac{m\pi}{p}x\sin\frac{\pi n}{q}y.$$

Образуем ряд

$$u(x,\ y,\ t)=\sum_{m,\,n=1}^{\infty}u_{m,\,n}(x,\ y,\ t).$$

Неизвестные коэффициенты $a_{m,\,n}$ определим из начального условия

$$\varphi(x,\ y)=\sum_{m,\,n=1}^{\infty}a_{m,\,n}\sin\frac{m\pi x}{p}\sin\frac{n\pi y}{q},$$

т. е. $a_{m,\,n}$ — коэффициенты двойного ряда Фурье:

$$a_{m,\,n}=\frac{4}{pq}\int_0^p\int_0^q\varphi(x,\ y)\sin\frac{m\pi x}{p}\sin\frac{n\pi y}{q}\,dxdy.$$

Окончательно решение поставленной задачи записывается в виде

$$u(x,\ y,\ t)=\sum_{m,\,n=1}^{\infty}\frac{4}{pq}\left(\int_0^p\int_0^q\varphi(x,\ y)\sin\frac{m\pi}{p}x\sin\frac{n\pi y}{q}\,dxdy\right)\times$$

$$\times\,\mathrm{e}^{-a^2\pi^2(m^2/p^2+n^2/q^2)t}\sin\frac{m\pi}{p}x\sin\frac{n\pi}{q}y.$$

Если $\varphi(x, y)$ представима в виде конечной суммы слагаемых вида $\sin\frac{m\pi}{p}x\sin\frac{n\pi}{q}y$, то аналитическое решение также сводится к конечной сумме элементарных функций. Методом Фурье, по аналогии с рассмотренной задачей, исследуются [31]: неоднородное уравнение теплопроводности; задачи распространения тепла в бесконечном цилиндре и цилиндре конечных размеров; задачи распространения тепла в однородном шаре и др.

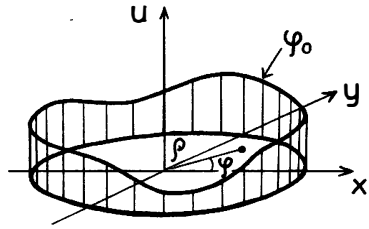

Рис. 5.8

7. Метод Фурье решения задачи Дирихле для уравнения Лапласа в круге. Рассмотрим уравнение (5.3.57) в круге

$$x^2+y^2<r_0^2$$

с условием

$$u|_\Gamma=\varphi_0, \tag{5.3.65}$$

где Г — граница круга (рис. 5.8).

Введем полярную систему координат (r, φ) и запишем уравнение Лапласа в полярных координатах

$$\Delta u=\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right)+\frac{1}{r^2}\frac{\partial^2 y}{\partial\varphi^2}=0. \tag{5.3.66}$$

Ищем частные решения (5.3.66) в виде

$$u=R(r)\Phi(\varphi).$$

Для определения функций $R(r)$, $\Phi(\varphi)$ получаем уравнения

$$\Phi''_{\varphi 2}+\lambda^2\Phi=0,$$

$$r\frac{d}{dr}\left(r\frac{dR}{dr}\right)-\lambda^2 R=0.$$

Общее решение первого уравнения

$$\Phi(\varphi)=c_1\cos\lambda\varphi+c_2\sin\lambda\varphi.$$

Функция $\Phi(\varphi)$ должна быть периодической с периодом 2π, т. е.

$$\Phi(\varphi+2\pi)=\Phi(\varphi).$$

Последнее равенство влечет условие

$$\lambda=n,$$

где n — целое число. Общее решение второго уравнения

$$R(r)=c_3r^n+c_4r^{-n}.$$

Поскольку ищется решение $u(r, \varphi)$ дважды непрерывно дифференцируемое в круге (т. е. и при $r=0$), следует положить $c_4=0$. Итак, частные решения

$$u_n(r, \varphi)=r^n(a_n\cos n\varphi+b_n\sin n\varphi), \quad 0\leqslant r\leqslant r_0, \quad 0\leqslant\varphi\leqslant 2\pi,$$

где a_n, b_n — произвольные константы. Образуем ряд

$$u(r, \varphi)=\sum_{n=0}^{\infty} u_n(r, \varphi);$$

неизвестные коэффициенты a_n, b_n определим из граничного условия $u(r_0, \varphi)=\varphi_0(\varphi)$, $0\leqslant\varphi\leqslant 2\pi$; φ_0 — заданная функция. Имеем

$$\varphi_0(\varphi)=\sum_{n=0}^{\infty} r_0^n(a_n\cos n\varphi+b_n\sin n\varphi).$$

Отсюда находим

$$a_0=\frac{1}{2\pi}\int_{-\pi}^{\pi}\varphi_0(\varphi)\,d\varphi, \quad a_n=\frac{1}{r_0^n\pi}\int_{-\pi}^{\pi}\varphi_0(\varphi)\cos n\varphi\,d\varphi,$$

$$b_n=\frac{1}{r_i^n\pi}\int_{-\pi}^{\pi}\varphi_0(\varphi)\sin n\varphi\,d\varphi.$$

Окончательное решение поставленной задачи имеет вид

$$u(r, \varphi)=\frac{1}{2\pi}\int_{-\pi}^{\pi}\varphi_0(\varphi)\,d\varphi+\sum_{n=1}^{\infty}\left(\frac{r}{r_0}\right)^n\left[\left(\frac{1}{\pi}\int_{-\pi}^{\pi}\varphi_0(\varphi)\cos n\varphi\,d\varphi\right)\cos n\varphi+\right.$$

$$\left.+\left(\frac{1}{\pi}\int_{-\pi}^{\pi}\varphi_0(\varphi)\sin n\varphi\,d\varphi\right)\sin n\varphi\right].$$

Последняя формула дает аналитическое решение для функций $\varphi_0(\varphi)$, представимых конечным отрезком ряда Фурье, а также в том случае, когда интегралы берутся в явном виде и ряд суммируется аналитически.

Приведем некоторые общие замечания относительно метода Фурье. Основная идея метода — свести решение уравнения с частными производными к системе обыкновенных дифференциальных уравнений, решение которой выражается в элементарных или специальных функциях. Однако для успешного применения аналитического метода необходимо, чтобы область пространственных переменных была ограничена координатными поверхностями тех систем координат, в которых переменные рассматриваемого уравнения разделяются. Именно поэтому выше рассмотрены прямоугольник и круг (более подробно см. [31]). Кроме того, аналитический метод эффективен в случае начальных и граничных условий, представляемых в виде конечной суммы по собственным функциям однородной задачи для данной области.

Если имеется возможность в технической задаче выбирать область, начальные и краевые условия, то следует принимать во внимание, что разумным их выбором можно получить аналитически решаемую задачу.

8. Метод подобия в нелинейной задаче теплопроводности. Рассмотрим уравнение теплопроводности или диффузии в том случае, когда коэффициент теплопроводности зависит от температуры, т. е. $k=k(u)$:

$$c\rho\frac{\partial u}{\partial t}=\frac{\partial}{\partial x}\left(k(u)\frac{\partial u}{\partial x}\right) \tag{5.3.67}$$

в области $0\leqslant x<\infty$, $0\leqslant t<\infty$ с начальным условием

$$u(x, 0)=u_0 \tag{5.3.68}$$

и граничным условием

$$u(0, t)=u_1, \tag{5.3.69}$$

где u_0, u_1 — константы. Это задача определения температуры тонкого полубесконечного стержня с начальной температурой u_0, к которому на границе ($x=0$) подключен тепловой резервуар, обеспечивающий на конце стержня температуру, равную u_1.

Заметим, что уравнение (5.3.67) не изменяется при преобразованиях независимых переменных:

$$x_1=qx, \quad t_1=q^2t,$$

т. е. остается подобным себе, если масштаб длин изменить в q раз, масштаб времени — в q^2 раз. При этих преобразованиях не изменяются и дополнительные условия (5.3.68), (5.3.69). Поэтому для решения поставленной задачи при любых x, t, q должно выполняться равенство

$$u(x, t)=u(qx, q^2t). \tag{5.3.70}$$

Положим $q=1/\sqrt{t}$. Тогда из (5.3.70) получим, что $u(x, t)$ — функция одной переменной $z=x/\sqrt{t}$

$$u(x, t)=u(x/\sqrt{t}, 1)=v(z).$$

Отсюда находим

$$\frac{\partial u}{\partial t}=\frac{dv}{dz}\frac{(-1)x}{2t^{3/2}}, \quad \frac{\partial u}{\partial x}=\frac{1}{t^{1/2}}\frac{dv}{dz}, \quad \frac{\partial}{\partial x}=\frac{1}{t^{1/2}}\frac{d}{dz}.$$

Подставим эти выражения в (5.3.67), получим уравнение

$$\frac{d}{dz}\left(k(v)\frac{dv}{dz}\right)=-c\rho\frac{1}{2}z\frac{dv}{dz} \tag{5.3.71}$$

с дополнительными условиями из (5.3.68), (5.3.69)

$$v(0)=u_1, \quad v(\infty)=u_0. \tag{5.3.72}$$

Таким образом, исходная задача для уравнения с частными производными свелась к краевой задаче для ОДУ. Аналитическое

решение задачи (5.3.71), (5.3.72) дает аналитическое решение исходной задачи.

● 5.4. Методы возмущений

В методах возмущений важную роль играют такие понятия, как асимптотические формулы и ряды; определим их предварительно.

5.4.1. Асимптотические формулы, оценки и ряды. Пусть функция $f(x)$ определена на множестве D изменения переменной x, пусть точка a — предельная точка D. Напомним понятие символов порядка o и O. Соотношения

$$\begin{aligned} &f(x)=o(\varphi(x)), \quad x\to a, \quad x\in D,\\ &f(x)\sim\varphi(x), \quad x\to a, \quad x\in D,\\ &f(x)=O(\varphi(x)), \quad x\in D,\\ &f(x)=O(\varphi(x)), \quad x\to a, \quad x\in D, \end{aligned} \tag{5.4.1}$$

эквивалентны следующим:

$$\begin{aligned} &f(x)/\varphi(x)\to 0, \quad x\to a, \quad x\in D,\\ &f(x)/\varphi(x)\to 1, \quad x\to a, \quad x\in D,\\ &|f(x)/\varphi(x)|\leqslant c<\infty, \quad x\in D,\\ &|f(x)/\varphi(x)|\leqslant c<\infty, \quad x\to a, \quad x\in D, \end{aligned}$$

где c — константа. Например, при $x\to 0$, $x>0$

$$\operatorname{sh}\frac{1}{x}=O\left(e^{1/x}\right),\ e^{-1/x}=o(1), \quad \sin x\sim x, \quad \cos x=O(1),$$

$$\ln\ln\frac{1}{x}=o\left(\ln\frac{1}{x}\right);$$

для комплексной переменной z

$$\frac{\sin z}{z}=o\left(e^{|\operatorname{Im} z|}\right), \quad z\to\infty, \quad 1-e^{-z}=O(z), \quad \operatorname{Re} z>0;$$

для натуральных чисел n

$$\ln n!\sim n\ln n, \quad n\to\infty.$$

Формулы (5.4.1) называют *асимптотическими формулами* или оценками.

Пусть имеется последовательность $\{u_n(x)\}$, $x\in D$, $n=0, 1, \ldots$, такая, что

$$\lim_{x\to a}\frac{u_{n+1}(x)}{u_n(x)}=0;$$

такая последовательность называется асимптотической.

Будем говорить, что ряд

$$\sum_{n=0}^{\infty} a_n u_n(x)$$

(не обязательно сходящийся) является *асимптотическим рядом* функции $f(x)$ по асимптотической последовательности $\{u_n(x)\}$, если для любого n

$$f(x)-\sum_{k=0}^{n} a_n u_n(x)=o(u_n(x)), \quad x\to a.$$

Хотя асимптотический ряд может быть расходящимся, его применение в вычислениях оказывается эффективным, поскольку можно фиксировать n и использовать его при $x\to a$. Таким образом, при фиксированном n вопрос о сходимости асимптотического ряда не имеет смысла, так как имеем конечную сумму n слагаемых.

Если асимптотическая последовательность фиксирована и асимптотическое разложение $f(x)$ существует, то оно единственно и его коэффициенты определяются формулой

$$a_n=\lim_{x\to a}\frac{f(x)-\sum_{k=0}^{n-1} a_k u_k(x)}{u_n}.$$

Этот факт записывается следующим образом:

$$f(x)\sim\sum_{n=0}^{\infty} a_n u_n(x).$$

Сравним характер поведения членов трех типов рядов: a) ряд сходится в теоретическом и практическом смысле; b) ряд сходится в теоретическом смысле и «расходится» в практическом; c) ряд сходится в теоретическом смысле, являясь асимптотическим (рис. 5.9).

Пример ряда a) для малых значений x

$$\operatorname{erf} x=\frac{2}{\sqrt{\pi}}\int_0^x e^{-s^2}ds=\frac{2}{\sqrt{\pi}}\sum_{n=0}^{\infty}(-1)^n\frac{x^{2n+1}}{n!(2n+1)}, \quad x^2<\infty. \qquad (5.4.2)$$

Пример ряда b): тот же ряд (5.4.2) для больших значений x, так как если x велико, то велико и число больших членов ряда. На практике этот ряд для вычислений $\operatorname{erf} x$ при больших

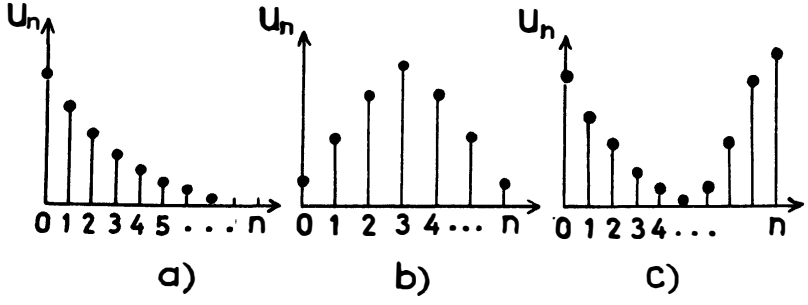

Рис. 5.9

значениях x не применяется. Более удобен следующий асимптотический ряд — пример ряда с):

$$\operatorname{erf} x \sim 1 - \frac{e^{-x^2}}{x\sqrt{\pi}}\left(1 - \frac{1}{2x^2} + \frac{1\cdot 3}{2^2 x^4} - \frac{1\cdot 3\cdot 5}{2^3 x^6} + \ldots\right). \tag{5.4.3}$$

Заметим, что ряд (5.4.3) расходится, но если оборвать ряд, например на втором члене, то получим формулу

$$\operatorname{erf} x = 1 - \frac{e^{-x^2}}{x\sqrt{\pi}}\left(1 - \frac{1}{2x^2}\right) + O\left(\frac{e^{-x^2}}{x^s}\right), \quad x \to \infty,$$

которую удобно использовать для достаточно больших x.

Впервые асимптотические ряды в вычисления ввел французский математик Пьер Лаплас (1749—1827). Современная теория асимптотических разложений берет начало с работ французского математика Анри Пуанкаре (1854—1912). Методы возмущений очень часто используют в качестве рядов по параметру возмущения ε асимптотические ряды

$$f(\varepsilon) \sim \sum_{n=0}^{\infty} a_n \varepsilon^n,$$

где асимптотическая последовательность — степенная $\{\varepsilon^n\}$, $\varepsilon \to 0$, $\varepsilon > 0$. Поэтому очень важно понимание свойств этих рядов.

Над асимптотическими рядами можно производить некоторые операции. Приведем основные свойства степенных асимптотических рядов:

Теорема 5.6. *Если*

$$f(\varepsilon) \sim \sum_{n=0}^{\infty} a_n \varepsilon^n, \quad g(\varepsilon) \sim \sum_{n=0}^{\infty} b_n \varepsilon^n, \quad \varepsilon \to 0, \quad \varepsilon > 0,$$

то

$$f(\varepsilon) + g(\varepsilon) \sim \sum_{n=0}^{\infty} c_n \varepsilon^n, \quad f(\varepsilon) g(\varepsilon) \sim \sum_{n=0}^{\infty} d_n \varepsilon^n, \quad \varepsilon \to 0, \quad \varepsilon > 0,$$

где $c_n = a_n + b_n$, $d_n = \sum_{k=0}^{n} a_k b_{n-k}$.

Доказательство. Докажем справедливость формулы суммирования. По определению имеем (для $N \geqslant 1$)

$$f(\varepsilon) = \sum_{n=0}^{N-1} a_n \varepsilon^n + O(\varepsilon^N), \quad g(\varepsilon) = \sum_{n=0}^{N-1} b_n \varepsilon^n + O(\varepsilon^N).$$

Складывая, находим

$$f(\varepsilon) + g(\varepsilon) = \sum_{n=0}^{N-1} (a_n + b_n) \varepsilon^n + O(\varepsilon^N),$$

отсюда, обозначая $c_n = a_n + b_n$, получаем формулу суммирования. Аналогично доказывается формула умножения.

Введем понятие равномерности асимптотических формул, оценок, рядов. Пусть в формулах (5.4.1) функция $f = f(x, y)$ зависит

еще от одной переменной y, пусть также $y \in D_1$. Тогда, если в соотношениях (5.4.1) предельный переход выполняется равномерно по y, а константу c можно выбрать не зависящей от y, будем говорить о равномерных асимптотических формулах, оценках для $f(x, y)$ по y. Например,

$$f(x, y) = o(1), \quad x \to a, \quad x \in D, \quad y \in D_1,$$

— равномерная оценка, если $f(x, y) \to 0$, $x \to a$, $x \in D$ равномерно по $y \in D_1$.

Приведем теорему об интегрируемости и дифференцируемости степенных асимптотических рядов, которая использует равномерность оценок.

Теорема 5.7. *Пусть*

$$f(\varepsilon, x) \sim \sum_{n=0}^{\infty} a_n(x) \varepsilon^n, \quad \varepsilon \to 0, \quad \varepsilon > 0, \quad a \leqslant x \leqslant b,$$

равномерно по x. *Тогда*

$$\int_a^b f(\varepsilon, x)\, dx \sim \sum_{n=0}^{\infty} \left(\int_a^b a_n(x)\, dx \right) \varepsilon^n, \quad \varepsilon \to 0, \quad \varepsilon \to 0 \qquad (5.4.4)$$

Пусть функция $f(\varepsilon, x)$ *дифференцируема по* x *и имеет место*

$$\frac{df(\varepsilon, x)}{dx} \sim \sum_{n=0}^{\infty} b_n(x) \varepsilon^n, \quad \varepsilon \to 0, \quad \varepsilon > 0, \quad a \leqslant x \leqslant b.$$

Тогда $b_n(x) = \dfrac{da_n(x)}{dx}$.

Рассмотрим пример

$$f(\varepsilon, x) = \sqrt{x + \varepsilon} = \sqrt{x} \left(1 + \frac{\varepsilon}{x} \right)^{1/2} = \sqrt{x} \left(1 + \frac{1}{2} \frac{\varepsilon}{x} + \frac{(1/2)(1/2)}{2} \left(\frac{\varepsilon}{x} \right)^2 + \ldots \right).$$

При $\varepsilon \to 0$ ряд в правой части равенства — равномерный асимптотический ряд на интервале $a \leqslant x \leqslant b$, где a не зависит от ε, $a > 0$, и неравномерный на интервале $0 < x < b$. Для неравномерного асимптотического ряда утверждение теоремы 5.7 может быть неверным, что можно проверить на приведенном примере.

5.4.2. Асимптотическое разложение интегралов, содержащих параметр. Простейший пример асимптотического разложения интеграла дает первая часть теоремы 5.7. — формула (5.4.4). Такое разложение можно трактовать как ряд возмущений по параметру возмущения ε, причем это регулярный ряд возмущений. Таким образом, если подынтегральная функция разлагается в равномерный по x асимптотический ряд при $\varepsilon \to 0$, $\varepsilon > 0$, то асимптотическое разложение получается почленным интегрированием этого ряда.

Например, для приведенного выше примера при $a > 0$ имеем

$$\int_a^b \sqrt{x + \varepsilon}\, dx = \left(x^{3/2} + \varepsilon x^{1/2} \right) \Big|_a^b + O(\varepsilon)^2, \quad \varepsilon \to 0, \quad \varepsilon > 0,$$

но интеграл

$$\int_0^b \sqrt{x+\varepsilon}\,dx=\frac{2}{3}(x+\varepsilon)^{3/2}\Big|_0^b=\frac{2}{3}(b+\varepsilon)^{3/2}-\frac{2}{3}\varepsilon^{3/2}=\frac{2}{3}b^{3/2}\left(1+\frac{\varepsilon}{b}\right)^{3/2}-$$

$$-\frac{2}{3}\varepsilon^{3/2}=\frac{2}{3}b^{3/2}\left(1+\frac{3}{2}\frac{\varepsilon}{b}+\frac{3/2\cdot 1/2}{2}\frac{\varepsilon^2}{b^2}+\ldots\right)-\frac{2}{3}\varepsilon^{3/2} \qquad (5.4.5)$$

уже имеет нерегулярный по ε член $\frac{2}{3}\varepsilon^{3/2}$ (дробную степень ε).

Этот пример свидетельствует, что важнейшим моментом в асимптотическом разложении интегралов по параметру ε является определение асимптотической последовательности $u_n(\varepsilon)$ такой, чтобы выполнялось соотношение

$$\int_a^b f(x,\varepsilon)\,dx\sim\sum_{n=0}^{\infty}a_n u_n(\varepsilon),\quad \varepsilon\to 0,\quad \varepsilon>0.$$

Из (5.4.5) следует, что $\{u_n(\varepsilon)\}=\{\varepsilon^0, \varepsilon^1, \varepsilon^{3/2}, \varepsilon^2, \ldots, \varepsilon^n, \ldots\}$. Часто не удается определить всю последовательность $\{u_n(\varepsilon)\}$ (а на практике это и не нужно) и тогда ограничиваются поиском первых ее членов $\{u_1(\varepsilon), u_2(\varepsilon), \ldots, u_N(\varepsilon)\}$ и доказательством соотношения

$$\int_a^b f(x,\varepsilon)\,dx=\sum_{n=0}^{N-1}a_n u_n(\varepsilon)+O\left(u_N(\varepsilon)\right),\quad \varepsilon\to 0.$$

Одним из простейших способов получения асимптотических разложений интегралов является *метод интегрирования по частям*. Найдем для примера асимптотику по ε следующего интеграла $F(\varepsilon)$ для $f(x)\in C^N[0,\infty)$:

$$F(\varepsilon)=\int_0^{\infty}e^{-x/\varepsilon}f(x)\,dx=-\varepsilon e^{-x/\varepsilon}f(x)\Big|_0^{\infty}+\varepsilon\int_0^{\infty}e^{-x/\varepsilon}f'(x)\,dx=$$

$$=\varepsilon f(0)-\varepsilon^2 e^{-x/\varepsilon}f'(x)\Big|_0^{\infty}+\varepsilon^2\int_0^{\infty}e^{-x/\varepsilon}f''(x)\,dx=\varepsilon f(0)+\varepsilon^2 f'(0)+$$

$$+\varepsilon^3 f''(0)+\ldots+\varepsilon^N f^{N-1}(0)+\varepsilon^N\int_0^{\infty}e^{-x/\varepsilon}f^N(x)\,dx.$$

Асимптотический характер получающегося ряда устанавливается оценкой остаточного члена

$$|R_N(\varepsilon)|=\varepsilon^N\left|\int_0^{\infty}e^{-x/\varepsilon}f^N(x)\,dx\right|\leqslant\varepsilon^N c\int_0^{\infty}e^{-x/\varepsilon}\,dx=c\varepsilon^{N+1},$$

где константа c оценивает $f^N(x)$: $|f^N(x)|\leqslant c$ на интервале $0\leqslant x<\infty$.

Рассмотрим *асимптотические разложения интегралов от быстроосциллирующих функций*. Численное интегрирование таких функций тем сложнее выполнить, чем выше частота осцилляций (см. гл. 7). Пусть $\varphi(x)\in C^N[a,b]$, тогда интеграл

$$F(\varepsilon)=\int_a^b e^{i\frac{x}{\varepsilon}}\varphi(x)\,dx \qquad (5.4.6)$$

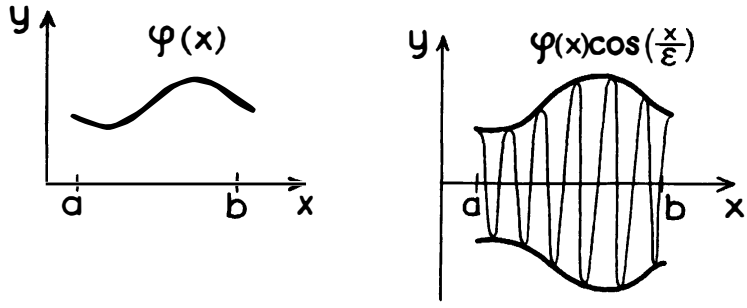

Рис. 5.10

(здесь i — мнимая единица) представляет собой интеграл от быстроосциллирующих функций (рис. 5.10):

$$\varphi(x)\cos(x/\varepsilon), \quad \varphi(x)\sin(x/\varepsilon), \quad \varepsilon\to 0, \quad \varepsilon>0.$$

Частота осцилляций $\omega=\omega(\varepsilon)$

$$\omega=(2\pi/\varepsilon)\to\infty; \quad \varepsilon\to 0.$$

По сравнению с численными методами асимптотические методы обладают свойством: чем выше частота $\omega(\varepsilon)$, тем меньше нужно брать членов асимптотического ряда для достижения одинаковой точности.

Этот ряд получается также интегрированием по частям. Имеем

$$F(\varepsilon)=\sum_{n=0}^{N-1} i^{n+1}\left[e^{i\frac{a}{\varepsilon}}\varphi^n(a)-e^{i\frac{b}{\varepsilon}}\varphi^n(b)\right]\varepsilon^{n+1}+R_N(\varepsilon),$$

где

$$R_N(\varepsilon)=i^N\varepsilon^N\int_a^b e^{i\frac{x}{\varepsilon}}\varphi^N(x)\,dx.$$

Так как функция $\varphi^N(x)$ непрерывна, то в силу леммы Римана

$$\int_a^b e^{i\frac{x}{\varepsilon}}\varphi^N(x)\,dx\to 0 \quad \text{при} \quad \varepsilon\to 0.$$

Таким образом,

$$R_N(\varepsilon)\sim o(\varepsilon^N).$$

Обобщением (5.4.6) является следующий интеграл:

$$F(\varepsilon)=\int_a^b e^{i\frac{f(x)}{\varepsilon}}\varphi(x)\,dx. \tag{5.4.7}$$

В (5.4.7) функция $\varphi(x)$ называется амплитудой, $\frac{1}{\varepsilon}f(x)$ — фазой подынтегральной функции. Вычисление асимптотического разложения $F(\varepsilon)$ при $\varepsilon\to 0$ производится *методом стационарной фазы*. Идея этого метода состоит в том, что главные члены асимптотики

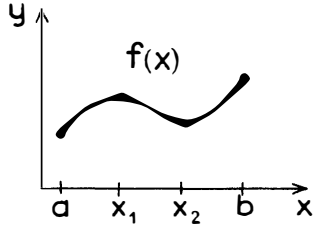

Рис. 5.11

$F(\varepsilon)$ определяются малыми окрестностями точек a, b и тех точек x_i интервала (a, b), где фаза $\frac{1}{\varepsilon}f(x)$ стационарна (рис. 5.11), т. е. $f'(x_i)=0$.

Рассмотрим конечное число стационарных точек x_i таких, что $f''(x_i)\neq 0$. Очевидно, что $[a, b]$ можно разделить на конеч-

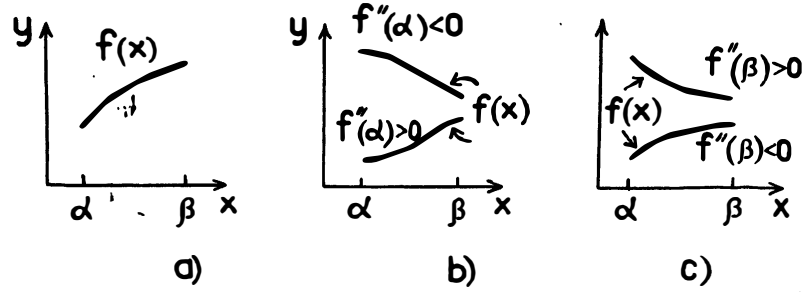

Рис. 5.12

ное число замкнутых интервалов, в каждом из которых $f'(x)$ или не обращается в нуль, или обращается в нуль в одном из концов этого интервала. Рассмотрим эти три случая отдельно (рис. 5.12).

Теорема 5.8. Случай a). *Пусть $f(x)$ не имеет стационарных точек на $[\alpha, \beta]$. Пусть $f(x)\in C^2[\alpha, \beta]$, $\varphi(x)\in C^1[\alpha, \beta]$. Тогда*

$$G(\varepsilon)=\int_\alpha^\beta e^{i\frac{f(x)}{\varepsilon}}\varphi(x)\,dx=\frac{\varepsilon}{i}\left[\frac{\varphi(\beta)}{f'(\beta)}e^{i\frac{f(\beta)}{\varepsilon}}-\frac{\varphi(\alpha)}{f'(\alpha)}e^{i\frac{f(\alpha)}{\varepsilon}}\right]+O(\varepsilon^2).$$

Случай b). *Пусть $f(x)$ имеет одну стационарную точку $x=\alpha$ на $[\alpha, \beta]$. Пусть $f(x)\in C^3[\alpha, \beta]$, $\varphi(x)\in C^1[\alpha, \beta]$. Тогда*

$$G(\varepsilon)=\int_\alpha^\beta e^{i\frac{f(x)}{\varepsilon}}\varphi(x)\,dx=\left[\frac{\pi\varepsilon}{\pm 2f''(\alpha)}\right]^{1/2}\varphi(\alpha)e^{i\frac{f(x)}{\varepsilon}\pm\frac{\pi i}{4}}+O(\varepsilon),$$

где знак плюс соответствует $f''(\alpha)>0$, знак минус — $f''(\alpha)<0$.

Случай c). Пусть $f(x)$ *имеет одну стационарную точку $x=\beta$ на $[\alpha, \beta]$. Пусть $f(x)\in C^3[\alpha, \beta]$, $\varphi(x)\in C^1[\alpha, \beta]$. Тогда*

$$G(\varepsilon)=\int_\alpha^\beta e^{i\frac{f(x)}{\varepsilon}}\varphi(x)\,dx=\left[\frac{\pi\varepsilon}{\pm f''(\beta)}\right]^{1/2}\varphi(\beta)e^{i\frac{f(x)}{\varepsilon}\pm\frac{\pi i}{4}}+O(\varepsilon),$$

где знак плюс соответствует $f''(\beta)>0$, знак минус — $f''(\beta)<0$.

Доказательство. Докажем наиболее простую часть теоремы — случай а). Интегрируя по частям, получаем

$$G(\varepsilon)=\frac{\varepsilon}{i}\int_{\alpha}^{\beta}\frac{\varphi}{f'}\frac{d}{dx}\left(e^{i\frac{f}{\varepsilon}}\right)dx=\frac{\varepsilon}{i}\left[\frac{\varphi(\beta)}{f'(\beta)}e^{i\frac{f(\beta)}{\varepsilon}}-\right.$$

$$\left.-\frac{\varphi(\alpha)}{f'(\alpha)}e^{i\frac{f(\alpha)}{\varepsilon}}\right]-\frac{\varepsilon}{i}\int_{\alpha}^{\beta}e^{i\frac{f}{\varepsilon}}\frac{d}{dx}\left(\frac{\varphi(x)}{f'(x)}\right)dx.$$

Очевидно, что $G(\varepsilon)$ имеет порядок $O(\varepsilon)$. Функция

$$\psi(x)=\frac{d}{dx}\left(\frac{\varphi(x)}{f'(x)}\right)$$

удовлетворяет тем же условиям, что и $\varphi(x)$, поэтому интеграл

$$\int_{\alpha}^{\beta}e^{i\frac{f}{\varepsilon}}\psi(x)\,dx$$

есть величина $O(\varepsilon)$, что и доказывает первую часть теоремы.

5.4.3. Теорема Пуанкаре. Как отмечалось в п. 5.1.4, имеется два основных типа рядов возмущений и соответственно два типа методов возмущений: регулярные и сингулярные. Теорема Пуанкаре относится к регулярным методам возмущений решений обыкновенных дифференциальных уравнений. Полученные ниже ряды возмущений по ε оказываются не только асимптотическими, но и сходящимися. Асимптотическая последовательность, используемая для построения рядов, степенная $\{\varepsilon^n\}$.

Рассмотрим систему дифференциальных уравнений

$$\frac{dy}{dx}=f(y, x, \varepsilon). \tag{5.4.8}$$

Здесь y — вектор $(y_1, y_2, \ldots, y_n)$, $a\leqslant x\leqslant b$, ε — параметр возмущения. Для (5.4.8) задаются начальные условия

$$y(a)=y^0. \tag{5.4.9}$$

Будем предполагать, что функция $f(y, x, \varepsilon)$ является аналитической функцией всех своих аргументов в области

$$D=\{\|y\|\leqslant d,\quad a\leqslant x\leqslant b,\quad 0\leqslant\varepsilon\leqslant\varepsilon_0\},$$

где $\|y^0\|\leqslant d$.

Рассмотрим невозмущенную систему уравнений, соответствующую (5.4.8),

$$\frac{dy}{dx}=f(y, x, 0) \tag{5.4.10}$$

с начальными условиями (5.4.9). Предположим, что решение (5.4.10), (5.4.9), $y_0(x)$ существует на интервале $a \leqslant x \leqslant b$, $\| y_0 \| \leqslant d$.

Будем искать решение (5.4.8), (5.4.9) в виде ряда

$$y(x, \varepsilon) = \sum_{k=0}^{\infty} y_k(x) \varepsilon^k. \tag{5.4.11}$$

Для определения $y_k(x)$, $k = 0, 1, 2 \ldots$, подставим (5.4.11) в (5.4.8) и, сравнивая коэффициенты при одинаковых степенях ε^k, получим системы уравнений для определения неизвестных вектор-функций

$$y_k(x) = (y_{k,1}(x), y_{k,2}(x), \ldots, y_{k,n}(x)), \quad k = 0, 1, 2, \ldots.$$

В нулевом приближении (ε^0) имеем

$$\frac{dy_0}{dx} = f(y_0, x, 0),$$

т. е. невозмущенную систему уравнений. Зададим для $y_0(x)$ начальное условие (5.4.9); тогда в силу единственности решения задачи Коши найдем, что $y_0(x)$ — невозмущенное решение (5.4.10), (5.4.9). В первом приближении (ε^1) имеем

$$\frac{dy_{1,i}}{dx} = \sum_{j=1}^{n} \frac{\partial f_i}{\partial y_j}(y_0(x), x, 0) y_{1,j} + \frac{\partial f_i}{\partial \varepsilon}(y_0(x), x, 0), \quad 1 \leqslant i \leqslant n. \tag{5.4.12}$$

В k-м приближении (ε^k, $k \geqslant 2$) имеем

$$\frac{dy_{k,i}}{dx} = \sum_{j=1}^{n} \frac{\partial f_i}{\partial y_j}(y_0(x), x, 0) y_{k,j} + \Phi_k(y_0(x), \ldots, y_{k-1}(x), x),$$

$$1 \leqslant i \leqslant n, \quad k = 2, 3, \ldots. \tag{5.4.13}$$

Функция $\Phi_k(y_0(x), \ldots y_{k-1}(x), x)$ получается разложением в ряды по y, ε функции $f(y, x, \varepsilon)$ и последующей подстановкой в них (5.4.11).

Для систем уравнений (5.4.12), (5.4.13) зададим нулевые начальные условия

$$y_k(a) = 0, \quad k = 1, 2, \ldots. \tag{5.4.14}$$

Заметим, что k-е приближения $k \geqslant 1$ определяются линейными системами ОДУ с одной и той же матрицей

$$\frac{\partial f_i}{\partial y_j}(y_0, x, 0), \quad 1 \leqslant i, \quad j \leqslant n,$$

и различными Φ_k, причем в k-ю функцию Φ_k входят только предыдущие приближения $y_0(x), \ldots, y_{k-1}(x)$. Таким образом, последовательно можем определить все $y_k(x)$ и ряд (5.4.11). Теорема Пуанкаре решает вопрос о сходимости полученного пока формального ряда (5.4.11); сформулируем ее без доказательства.

Теорема 5.9. *Пусть $f(y, x, \varepsilon)$ — аналитическая функция своих аргументов в области D. Пусть также невозмущенное решение*

$y_0(x) \in D$. *Тогда ряд* (5.4.11) *сходится при достаточно малых* ε $(0<\varepsilon<\varepsilon_0)$ *к решению* (5.4.8), (5.4.9) *и имеет место равенство*

$$y(x, \varepsilon)=\sum_{k=0}^{N-1} y_k(x)\varepsilon^k+O(\varepsilon^N), \quad \varepsilon\to 0, \quad \varepsilon>0.$$

Покажем, что в том случае, когда известен общий интеграл (5.4.10), т. е.

$$y_i(x)=F_i(x, c_1, \ldots, c_n), \quad 1\leqslant i\leqslant n, \tag{5.4.15}$$

где c_i — произвольные постоянные, задача определения любых $y_k(x)$, $k\geqslant 1$, сводится к процедуре дифференцирования функций F_i и интегрированию.

Матрица

$$R(x)=r_{i,j}(x)=\frac{\partial F_i(x, c)}{\partial c_j}$$

определяет фундаментальную матрицу решений однородной системы, соответствующей (5.4.12), (5.4.13). Действительно, имеем

$$\frac{dr_{i,j}}{dx}=\frac{d}{dx}\frac{\partial F_i}{\partial c_j}=\frac{\partial}{\partial c_j}\frac{d}{dx}F_i=\frac{\partial}{\partial c_j}f_i(F(x, c), x, 0)=$$

$$=\sum_{k=1}^{n}\frac{\partial f_i}{\partial F_k}\frac{\partial F_k}{\partial c_j}=\sum_{k=1}^{n}\frac{\partial f_i}{\partial y_k}(F, x, 0)r_{k,j}, \quad 1\leqslant j\leqslant n.$$

Пусть выбраны константы c_i^0 так, чтобы

$$y_i^0=F_i(a, c_1^0, \ldots, c_n^0);$$

тогда

$$\frac{dr_{i,j}}{dx}=\sum_{k=1}^{n}\frac{\partial f_i}{\partial y_k}(y_0(x), x, 0)r_{k,j}.$$

При этом можно показать, что

$$\det\frac{\partial F_i}{\partial c_j}(a, c_1^0, \ldots, c_n^0)\neq 0$$

и, следовательно, $R(x)$ — фундаментальная матрица решений однородной системы (5.4.12), (5.4.13).

Теперь, используя $R(x)$, можно записать решения для $y_k(x)$, $k\geqslant 1$. Например, для $k\geqslant 2$ получаем

$$y_k(x)=R(x)\int_1^x R^{-1}(s)\Phi_k(y_0(s), \ldots, y_{k-1}(s))\,ds. \tag{5.4.16}$$

Таким образом, действительно, определение $y_k(x)$ сводится к процедуре дифференцирования F_i для определения $R(x)$ и интегрирования по формулам вида (5.4.16).

В качестве примера рассмотрим уравнение колебаний тела массы m

$$m\frac{d^2y}{dx^2}+\alpha y+\beta y^3=0, \quad 0<x<b,$$

притягиваемой к положению равновесия упругой силой $\alpha y+\beta y^3$, $\alpha>0$, $\beta>0$, начальные условия имеют вид

$$y(0)=y^0, \quad \frac{dy}{dx}(0)=0; \tag{5.4.17}$$

переменная x имеет смысл времени. Обозначим $\omega^2=\alpha/m$, $\varepsilon=\beta/m$. Исходное уравнение приведем к виду

$$\frac{d^2y}{dx^2}+\omega^2 y+\varepsilon y^3=0 \tag{5.4.18}$$

и будем искать асимптотику задачи (5.4.18), (5.4.17) в виде ряда по ε:

$$y(x, \varepsilon)=\sum_{k=0}^{\infty} y_k(x)\varepsilon^k.$$

Запишем приближение с точностью до $O(\varepsilon^2)$. Имеем

$$\frac{d^2y_0}{dx^2}+\omega^2 y_0=0, \quad y_0(0)=y^0, \quad \frac{dy_0}{dx}(0)=0,$$

$$\frac{d^2y_1}{dx}+\omega^2 y_1=-y_0^3, \quad y_1(0)=0, \quad \frac{dy_1}{dx}(0)=0,$$

откуда находим

$$y_0(x)=y^0\cos\omega x,$$

$$y_1(x)=-\frac{3}{8\omega}(y_0)^3 x\sin\omega x+\frac{(y_0)^3}{32\omega^2}(\cos 3\omega x-\cos\omega x).$$

Следовательно,

$$y(x, \varepsilon)=y^0\cos\omega x+\varepsilon\left[\frac{(y_0)^3}{32\omega^2}(\cos 3\omega x-\cos\omega x)-\right.$$

$$\left.-\frac{3(y_0)^3}{8\omega}x\sin\omega x\right]+O(\varepsilon^2). \tag{5.4.19}$$

Формула (5.4.19) содержит слагаемое

$$-\varepsilon\frac{3(y_0)^3}{8\omega}x\sin\omega x,$$

которое соответствует бесконечному росту (при увеличении x) амплитуды колебания и находится в противоречии с законом сохранения энергии

$$\frac{1}{2}m\left(\frac{dy}{dx}\right)^2+\frac{1}{2}\alpha y^2+\frac{\beta}{4}y^4=\text{const},$$

поскольку из этого равенства следует, что

$$y^2<2\text{const}/\alpha.$$

Таким образом, формула (5.4.19) справедлива только на интервале $0<x\leqslant b$, длина b которого не зависит от ε.

На интервалах длины порядка $O(1/\varepsilon)$ уже применяются иные асимптотические разложения, нежели (5.4.11), и отличные от метода Пуанкаре асимптотические методы [19].

● 5.5. Численные методы

Рассмотрим некоторые общие вопросы численного анализа. Численные методы в конкретных классах математических задач изучаются в гл. 6—11.

Как уже отмечалось в **5.1**, численным мы называем метод решения задачи, который сводится только к арифметическим действиям над числами. В этом состоит их существенное отличие от аналитических методов и методов возмущений, которые могут иметь дело с объектами, отличными от чисел, например с функциями.

Заметим, что аналитические методы после этапа манипулирования с формулами на этапе получения численного результата также сводятся к численным методам. Действительно, если результат представлен в виде конечного числа или бесконечной последовательности формул, а необходимо получить числовое значение результата, то обязательно приходим к этапу арифметических действий над числами. После аналитического представления решения задачи (получение формул) следует провести вычисления с числами, оценить погрешность результата, а это уже область действия численного метода. Таким образом аналитические методы ведут к численным. Поэтому роль численных методов в вычислительной математике столь значительна.

5.5.1. Реализация численных методов. Так как в ЭВМ числа записываются конечным набором двоичных разрядов, то можно сделать следующие выводы:

1) на ЭВМ существует возможность представить лишь ограниченное количество чисел;

2) на различных ЭВМ множество этих чисел может быть различным из-за отличия количества разрядов и формы представления числа.

На ЭВМ с двоичным представлением чисел есть такие числа, как 1/2, 1/8, но нет таких, как 1/3, 2/5. Эти числа заменяются в вычислениях приближенно теми близкими числами, которые могут быть представлены в ЭВМ.

Кроме того, в различных ЭВМ может быть по-разному реализована арифметика чисел, например округление после выполнения операций, что также следует учитывать при реализации численных методов.

Таким образом, имеет смысл говорить о *теоретическом численном методе* и *реализации численного метода.*

Теоретический численный метод строится на бесконечных множествах целых и вещественных чисел и арифметике, его реализация основана на конечном множестве чисел ЭВМ и арифметике ЭВМ.

В дальнейшем будем рассматривать теоретические численные методы, опуская при этом термин «теоретические». Особенности, которые появляются при реализации численных методов, обсуждаются в **5.6.**

5.5.2. Прямая и обратная вычислительная задачи. В математической постановке задач можно выделить два класса, к которым относятся многие из рассматриваемых задач в численном анализе.

1) Прямая вычислительная задача: по заданному элементу y, принадлежащему некоторому пространству Y и оператору F, переводящему элементы Y в элементы пространства Z, найти

$$z=F(y). \tag{5.5.1}$$

Например, нахождение интеграла от $y(x)\in C[0, 1]$ — прямая вычислительная задача. Здесь

$$F(y)=\int_0^1 y(x)dx,$$

z — это число, $Y=C[0, 1]$, $Z=R^1$.

2) Обратная вычислительная задача: по заданному элементу $z\in Z$ и оператору F, переводящему элементы Y в элементы Z, найти такое y, чтобы выполнялось (5.5.1), т. е. следует решить уравнение (5.5.1).

Например, решение уравнения

$$z(x)=y(x)+\int_0^x y(s)ds, \quad 0\leqslant x\leqslant 1, \tag{5.5.2}$$

при заданной функции $z(x)\in C[0, 1]$, т. е. определение $y(x)\in C[0, 1]$ — обратная вычислительная задача. Здесь

$$F(y)=y-\int_0^x y(s)ds, \quad Y=Z=C[0, 1].$$

Заметим, что если известен обратный к F оператор $F^{-1}(z)=y$, то обратная задача является прямой вычислительной задачей.

Например, в задаче (5.5.2)

$$y(x)=z(x)-e^{-x}\int_0^x e^s z(s)ds=F^{-1}(z). \tag{5.5.3}$$

В том случае, когда F^{-1} неизвестен, решение обратной задачи состоит в построении F^{-1} или его приближении с помощью комбинации прямых вычислительных задач. Например, если бы мы не знали явный вид обратного оператора в (5.5.2), то приближения к F^{-1} можно найти методом последовательных приближений:

$$y_{n+1}(x)=z(x)-\int_0^x y_n(s)ds, \quad n=0, 1...; \quad y_0(x)=z(x),$$

откуда

$$y_1(x)=z(x)-\int_0^x z(s)ds,$$

$$y_2(x)=z(x)-\int_0^x z(s)ds+\int_0^x\int_0^s z(s_1)ds_1 ds,$$

.....

$$y_n(x)=z(x)-\int_0^x z(s)ds+\int_0^x\int_0^s z(s_1)ds_1 ds+\ldots+(-1)^{n-1}\int_0^x\ldots\int_0^{s_{n-2}} z(s_{n-1})ds_{n-1}.$$

Меняя порядок интегрирования, находим

$$\int_0^x\left(\int_0^x z(s_1)ds_1\right)ds=\int_0^x z(s_1)\left(\int_{s_1}^x ds\right)ds_1=\int_0^x(x-s)z(s)ds,$$

$$\int_0^x\int_0^s\ldots\int_0^{s_{n-2}} z(s_{n-1})ds_{n-1}ds_{n-2}\ldots ds=\int_0^x\frac{(x-s)^{n-1}}{(n-1)!}z(s)ds.$$

Следовательно,

$$y_n(x)=z(x)-\int_0^x\sum_{k=0}^{n-1-1^k}\frac{(x-s)^k}{k!}z(s)ds=\Phi_n(z). \qquad (5.5.4)$$

Таким образом, обратная вычислительная задача свелась к последовательности ($n=1, 2, \ldots$) прямых вычислительных задач — интегрированию по формулам (5.5.4). Одновременно мы построили приближения к оператору F^{-1}, а именно Φ_n, а также приближения $y_n(x)$ к точному решению (5.5.2).

Если заданная функция $z(x)$ такова, что интегралы в (5.5.3) или (5.5.4) аналитически не берутся, то применяется численный метод интегрирования.

Вообще в численных методах решения обратных вычислительных задач обязательно присутствует этап сведения к прямым вычислительным задачам: одной или последовательности. Способы сведения могут быть различными, что порождает и разнообразие численных методов в конкретных задачах.

5.5.3. Дискретизация в непрерывной задаче. Многие вычислительные задачи, прямые и обратные, имеют в качестве элементов y, z в (5.5.1) элементы бесконечномерных пространств Y, Z, например функции $y(x)$, $z(x)\in C[0, 1]$, как в примере (5.5.2). Такие задачи будем называть непрерывными в отличие от дискретных, где y, z в (5.5.1) — элементы конечномерных пространств. К дискретным задачам можно отнести вычислительные задачи линейной алгебры. Например, решение системы линейных уравнений

$$Ay=z,$$

где векторы $y=(y_1, \ldots, y_n)$, $z=(z_1, \ldots, z_n)$, матрица $A=(a_{i,j})$, $1 \leqslant i, j \leqslant n$,— дискретная задача.

Аналитические методы и методы возмущений могут оперировать с такими объектами, как функции. Численные же методы, если предполагается их реализация на ЭВМ, не могут оперировать с объектами, для описания которых требуется бесконечный набор чисел из-за конечного запаса чисел ЭВМ.

Действительно, функция $y(x)$, отличная от полинома, разлагающаяся в ряд Тейлора с центром в точке $x=a$, т. е.

$$y(x)=y(a)+\frac{y'(a)}{1!}(x-a)+\frac{y''(a)}{2!}(x-a)^2+\ldots+\frac{y^n(a)}{h!}(x-a)^n+\ldots,$$

требует для своего точного описания бесконечного числа значений $\{y(a), y'(a), y''(a), \ldots, y^n(a), \ldots\}$ и не может быть представлена конечным набором чисел. Поэтому важнейшим элементом численного анализа непрерывных задач является замена исходной задачи (5.5.1) последовательностью других, в некотором смысле «близких» дискретных задач

$$\{z_1=F_1(y_1)\}_i, \quad i=1, 2, \ldots. \tag{5.5.5}$$

Процедура перехода от (5.5.1) к (5.5.5) называется *дискретизацией* непрерывной задачи. Замена должна быть такой, чтобы решения (5.5.5) аппроксимировали в некотором смысле, который следует определить, точное решение (5.5.1) тем точнее, чем больше членов последовательности берется.

Заметим, что в предложенном подходе к численному анализу содержится предположение: исходная задача, если она обратная, имеет решение, т. е. вопросы существования, единственности решения (5.5.1) мы не затрагиваем, а занимаемся построением последовательностей $\{z_1, F_1, y_1\}_i$, которые дают сходящиеся по некоторой норме решения (5.5.5) к точному решению (5.5.1).

Методы дискретизации непрерывных задач весьма разнообразны. Некоторые из них рассматриваются в гл. 6—11.

Для примера рассмотрим один из способов перехода к дискретной задаче в уравнении (5.5.2). Разобьем интервал $0 \leqslant x \leqslant 1$ на N интервалов точками $x_i=ih$, $0 \leqslant i \leqslant N$, $h=1/N$. Вычислив в точках x_i функцию $z(x)$, получим вектор $z(x_i)$, $0 \leqslant i \leqslant N$.

По теореме Вейерштрасса непрерывную функцию $y(x)$ можно с любой точностью приблизить полиномом N-й степени при $N \to \infty$ в норме пространства $C[0, 1]$. Поэтому есть надежда, что если в (5.5.2) выполнить замену $y(x) \to \sum_{i=0}^{N} c_i x^i$, само уравнение заменить системой равенств

$$z(x_j)=\sum_{i=0}^{N} c_i x_j^i+\sum_{i=0}^{N}\int_0^{x_j} c_i s^i ds, \quad 0 \leqslant j \leqslant N, \tag{5.5.6}$$

решить систему линейных уравнений (5.5.6), т. е. найти коэффициенты $(c_0, c_1, \ldots, c_N)$, то при $N \to \infty$ полином

$$P_N(x) = \sum_{i=0}^{N} c_i x^i$$

будет равномерно сходиться к точному решению (5.5.2), а именно

$$\max_{0 \leqslant x \leqslant 1} |P_N(x) - y(x)| \to 0, \quad N \to \infty. \tag{5.5.7}$$

Переход от (5.5.2) к (5.5.6) и есть дискретизация непрерывной задачи.

Сравнивая (5.5.6) с (5.5.5), находим, что $z_1 = (z(x_i))$, $y_1 = P_N(x)$, $0 \leqslant i \leqslant N$. Оператор F_1 представляется матрицей с элементами

$$\left(x_j^i + \frac{x^{i+1}}{i+1}\right), \quad 0 \leqslant i \leqslant N.$$

В развернутой форме (5.5.6) имеет вид

$$z(0) = c_0,$$

$$z(h) = c_0 + c_1 h + \ldots + c_N h^N + c_0 h + c_1 \frac{h^2}{2} + \ldots + c_N \frac{h^{N+1}}{N+1},$$

$$z(2h) = c_0 + c_1 2h + \ldots + c_N (2h)^N + c_0 (2h) + c_1 \frac{(2h)^2}{2} + \ldots + c_N \frac{(2h)^{N+1}}{N+1},$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$z(Nh) = c_0 + c_1 (Nh) + \ldots + c_N (Nh)^N + c_0 (Nh) + c_1 \frac{(Nh)^2}{2} + \ldots + c_N \frac{(Nh)^{N+1}}{N+1}.$$

Обоснование сходимости приближений (доказательство (5.5.7)), оценка скорости сходимости (зависимость от N), обоснование разрешимости системы линейных уравнений (5.5.6) и определение способа решения (5.5.6) — вот круг вопросов, которые решаются в рамках численного анализа.

Как правило, способ перехода от непрерывной задачи к дискретной и отличает один численный метод от другого в одном и том же классе задач.

● 5.6. Оценка результатов вычислений

Обращая внимание на схему решения технической задачи (см. рис. В.1), замечаем, что результаты вычислений на ЭВМ необходимо оценить с точки зрения поставленной задачи. В первую очередь необходимо иметь оценку допущенной погрешности в вычислениях. Роль этого этапа в вычислительном процессе сильно зависит от характера принимаемых решений в технической задаче на основе результатов вычислений. Если ошибка в третьем знаке после запятой некоторого числа может привести к тому, что

вычислитель «взлетит на воздух», то, по-видимому, он должен уделить большое внимание оценкам возможных погрешностей и провести вычисления с пятью верными знаками.

Оценка погрешности, которая может быть получена до проведения вычислений, называется *априорной*, после проведения — *апостериорной*.

5.6.1. Три типа погрешностей. В процессе решения задачи возникают следующие погрешности:

1) погрешность математической модели;
2) погрешность численного метода;
3) погрешность вычислений на ЭВМ.

Рассмотрим три типа погрешностей на примере. Предположим, что решается задача определения одномерного движения материальной точки массы m, которая сводится к интегрированию уравнения Ньютона

$$m\frac{d^2y}{dx^2}=f\left(\frac{dy}{dx},\ y,\ x\right)+\varepsilon\left(\frac{dy}{dx},\ y,\ x\right),\quad a\leqslant x\leqslant b, \tag{5.6.1}$$

с начальными условиями

$$y(a)=y_0,\quad \frac{dy}{dx}(a)=y_1; \tag{5.6.2}$$

здесь y — координата точки, x — время.

В правой части (5.6.1) сила, действующая на тело, которое идеализированно заменено точкой, разделена на два слагаемых: f и ε. Первое слагаемое — это учитываемые силы, второе — неучитываемые, они фактически отбрасываются, и математическая модель описывается уравнением

$$m\frac{d^2y}{dx^2}=f\left(\frac{dy}{dx},\ y,\ x\right). \tag{5.6.3}$$

Если считать, что реальный процесс $y(x)$ дает решение задачи (5.6.1), (5.6.2), то, обозначая решение модельной задачи (5.6.3), (5.6.2) через $y_1(x)$, получаем

$$\delta_1(x)=y(x)-y_1(x),\quad a\leqslant x\leqslant b,$$

где $\delta_1(x)$ — погрешность математической модели.

Предположим, что точное решение задачи (5.6.3), (5.6.2) представляется сходящимся степенным рядом

$$y_1(x)=\sum_{k=0}^{\infty}A_k(x-a)^k, \tag{5.6.4}$$

где коэффициенты A_k определяются функцией f и могут быть найдены для любого k по известным формулам.

В качестве численного метода предложим нахождение A_k, $1\leqslant k\leqslant N$, по этим формулам и определение N-й частичной суммы ряда (5.6.4) (дискретизация)

$$y_2(x)=\sum_{k=0}^{N}A_k(x-a)^k. \tag{5.6.5}$$

Таким образом появляется погрешность

$$\delta_2(x)=y_1(x)-y_2(x),$$

где $\delta_2(x)$ — погрешность численного метода.

Если в задаче требуется найти значение $y(x)$ — положение тела в точке x, то следует реализовать вычисления A_k, суммирование и возведение в степень на ЭВМ по формулам

$$y_3(x^*)=\sum_{k=0}^{N} A_k^* \otimes (x^* \ominus a^*)^k. \qquad (5.6.6)$$

В (5.6.6) звездочкой отмечены числа ЭВМ, которые, как уже известно, записываются с округлением в конечное число разрядов и отличаются от точных в (5.6.5), хотя и близки к ним. Знаки $\otimes$, $\ominus$ соответствуют арифметическим операциям $\times$, $-$, реализованным на ЭВМ. Результаты арифметических операций ЭВМ также могут отличаться от точных. Эти вопросы подробнее рассматриваются в п. 5.6.5.

Сравнивая (5.6.6) с (5.6.5), находим

$$\delta_3(x)=y_2(x)-y_3(x^*),$$

где $\delta_3(x)$ — погрешность вычислений на ЭВМ.

Полная погрешность $\delta(x)$ численного решения задачи на ЭВМ — это сумма трех указанных выше погрешностей:

$$\delta(x)=\delta_1(x)+\delta_2(x)+\delta_3(x)=y(x)-\sum_{k=0}^{N} A_k^* \otimes (x^* \ominus a^*)^k.$$

Чтобы оценить погрешность, вводят какую-либо норму для функций $\delta_i(x)$, например равномерную $C[a, b]$, затем оценивают каждую погрешность отдельно и используют неравенство

$$\|\delta(x)\| \leqslant \|\delta_1(x)\| + \|\delta_2(x)\| + \|\delta_3(x)\|. \qquad (5.6.7)$$

Для δ_1 обычно находят априорную оценку, используя специальные технические знания. Получение оценки δ_2, априорной или апостериорной,— одна из основных задач создания методов вычислений и является составной частью метода. Оценка δ_3, как правило, получается апостериорно.

Из (5.6.7) вытекает следующая практическая рекомендация: при решении задачи величины погрешностей (нормы) трех типов δ_1, δ_2, δ_3 желательно, чтобы были одного порядка. Действительно, значительная разница в δ_1 и δ_2, а именно $\|\delta_2\| \ll \|\delta_1\|$, может привести к бессмысленной трате машинного времени, так как точность решения задачи лимитирует величина $\|\delta_1\|$.

Несколько сложнее обстоит дело с выполнением соотношения $\|\delta_3\| \sim \|\delta_2\|$. В некоторых вычислениях оказывается, что $\|\delta_3\| \ll \ll \|\delta_2\|$ из-за характеристик ЭВМ, имеющих большое число разрядов для хранения чисел, а следовательно, приводящих к избыточной точности вычислений. Создается впечатление, что повышенная точность вычислений «дается даром», а ситуацию нельзя

изменить. Но это не так. Лишние разряды машинного слова могут использоваться для более плотной упаковки в памяти ЭВМ чисел или перехода к целочисленной арифметике — все эти операции приводят к ускорению вычислений, но техника таких вычислений выходит за рамки настоящей книги.

Если оказывается, что $\|\delta_3\| > \|\delta_2\|$, то обычно переходят к вычислениям с двойной точностью, т. е. к использованию описания чисел типа DOUBLE PRECISION. Это дает возможность в представлении чисел держать ~ 17 десятичных разрядов после запятой, но при этом увеличивается время вычислений.

5.6.2. Абсолютная и относительная погрешности чисел. Результатом вычислений на ЭВМ являются числа. Поэтому погрешность вычислений — это погрешность чисел, полученных в результате вычислительного процесса.

Для записи чисел и указания их погрешности часто применяются понятия абсолютной и относительной погрешностей. Пусть a — точное значение некоторой величины, a_* — приближение к a.

Абсолютной погрешностью приближения a_* называется величина $\Delta(a_*)$, удовлетворяющая неравенству

$$|a_* - a| \leqslant \Delta(a_*).$$

Например, $a=\pi$, $a_*=3{,}14$; тогда

$$|3{,}14-\pi| \leqslant 0{,}002 = \Delta(3{,}14);$$

пусть $a_*=3{,}141$, тогда

$$|3{,}141-\pi| \leqslant 0{,}0006 = \Delta(3{,}141).$$

Относительной погрешностью приближения a_* называется величина $\delta(a_*)$, удовлетворяющая неравенству

$$\left|\frac{a_*-a}{a_*}\right| \leqslant \delta(a_*).$$

В качестве $\delta(a_*)$ можно взять

$$\delta(a_*)=\frac{\Delta(a_*)}{|a_*|}.$$

Имеем

$$\delta(3{,}14)=\frac{0{,}002}{3{,}14} \simeq 0{,}064\%,$$

$$\delta(3{,}141)=\frac{0{,}0006}{3{,}141} \simeq 0{,}002\%.$$

Абсолютная и относительная погрешности записываются в виде чисел с одной или двумя значащими цифрами, при этом производится округление с избытком. Абсолютная и относительная погрешности указываются в записи чисел следующим образом:

$$a = a_* \pm \Delta; \quad a = a_*(1 \pm \delta).$$

Например,

$$\pi = 3{,}14 + 0{,}002; \quad \pi = 3{,}141(1 + 0{,}002\%).$$

Значащие цифры десятичного числа — это все его цифры начиная с первой ненулевой слева.

Например, в записи числа 0,002306 подчеркнуты значащие цифры.

Значащая цифра в записи числа верна, если абсолютная погрешность числа меньше или равна пяти единицам разряда, следующего за этой цифрой.

Например, в записи чисел

$$0{,}00\underline{2}306 \pm 0{,}00001;\quad 0{,}00\underline{2}306 \pm 0{,}00006$$

подчеркнуты верные значащие цифры.

5.6.3. Потеря верных значащих цифр при вычитании близких чисел. Это явление легко продемонстрировать на примере. Пусть имеем два близких числа

$$x_1 = 0{,}0\underline{1234},\quad x_2 = 0{,}0\underline{1231}$$

с абсолютной погрешностью 0,000002 (подчеркнуты верные значащие цифры). Разность чисел

$$y = x_1 - x_2 = 0{,}0000\underline{3}$$

имеет только одну верную значащую цифру с абсолютной погрешностью 0,000004 (см. п. 5.6.4). Произошла потеря трех верных значащих цифр. Этот факт следует учитывать при составлении алгоритма. Можно пытаться избежать потери точности с помощью алгебраических преобразований.

Например, решение квадратного уравнения

$$x^2 + px + q = 0$$

при малых q

$$x_{1,2} = -p/2 \pm \sqrt{p^2/4 - q}$$

следует представить в эквивалентном виде

$$x_1 = -p/2 - \sqrt{p^2/4 - q},\quad x_2 = \frac{-q}{p/2 + \sqrt{p^2/4 - q}},$$

чтобы в вычислениях не содержалось вычитание близких чисел $p/2$ и $\sqrt{p/4 - q}$.

Изменение последовательности вычислений также может привести к повышению точности результата. Например, расстановка скобок в выражении

$$y = (a - b) + c = a + (c - b)$$

во втором варианте предпочтительнее при близких по значению числах a и b.

5.6.4. Оценка погрешности функции по погрешности аргументов. Пусть для простоты изложения имеется функция двух переменных

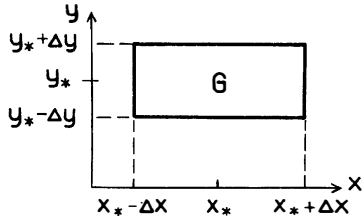

Рис. 5.13

$u(x, y)$, значение которой вычисляется в точке (x, y), заданной приближениями (x_*, y_*), а именно

$$x=x_*\pm\Delta x,\quad y=y_*\pm\Delta y,$$

т. е. аргументы задаются с абсолютной погрешностью Δx, Δy, а следовательно, областью неопределенности точных значений (x, y) является область $G=\{(x, y): x_*-\Delta x\leqslant x\leqslant x_*+\Delta x,\ y_*-\Delta y\leqslant y\leqslant y_*+\Delta y\}$ (рис. 5.13). Требуется оценить абсолютную погрешность функции

$$\Delta u(x_*, y_*),$$

удовлетворяющую неравенству

$$|u(x, y)-u(x_*, y_*)|\leqslant\Delta u(x_*, y_*).$$

Теорема 5.10. *Пусть функция $u(x, y)$ непрерывна вместе со своими первыми производными по x, y в области G. Тогда*

$$\Delta u(x_*, y_*)=\max_G\left|\frac{\partial u}{\partial x}(x, y)\right|\Delta x+\max_G\left|\frac{\partial u}{\partial y}(x, y)\right|\Delta y.$$

Доказательство. Представим функцию $u(x, y)$ формулой Тейлора

$$u(x, y)=u(x_*, y_*)+\frac{\partial u}{\partial x}(x_*+\theta(x-x_*),\quad y_*+\theta(y-y_*))$$

$$(x-x_*)+\frac{\partial u}{\partial y}(x_*+\theta(x-x_*),\quad y_*+\theta(y-y_*))(y-y_*),$$

где $0<\theta<1$. Переходя в этой формуле к оценкам, получим

$$|u(x, y)-u(x_*, y_*)|\leqslant\max_G\left|\frac{\partial u}{\partial x}\right||x-x_*|+$$

$$+\max_G\left|\frac{\partial u}{\partial y}\right||y-y_*|\leqslant\max_G\left|\frac{\partial u}{\partial x}\right|\Delta x+\max_G\left|\frac{\partial u}{\partial y}\right|\Delta y,$$

что и требовалось доказать.

Из теоремы 5.10 следует, что для функций $u(x, y)=x\pm y$ абсолютная погрешность $\Delta u=\Delta x+\Delta y$, т. е. *абсолютная погрешность суммы и разности двух чисел равна сумме абсолютных погрешностей этих чисел.*

Вычислим относительную погрешность функции:

$$u(x, y)=xy.$$

Из теоремы 5.10 находим

$$\delta u=\frac{\Delta u}{|x_*y_*|}=\frac{\max\limits_G|y|\,\Delta x+\max\limits_G|x|\,\Delta y}{|x_*y_*|}=\frac{(|y_*|+\Delta y)\Delta x+(|x_*|+\Delta x)\Delta y}{|x_*y_*|}=$$

$$=\delta x+\delta y+2\delta x\delta y,$$

т. е. *относительная погрешность произведения двух чисел с точностью до членов второго порядка малости равна сумме относительных погрешностей этих чисел,* а именно

$$\delta u \simeq \delta x + \delta y.$$

5.6.5. Влияние ошибок округления. Большинство задач вычислительной математики сводятся к четырем арифметическим операциям над целыми и вещественными числами.

Заметим, что сложение, вычитание и умножение целых чисел выполняется точно. Деление целых чисел осуществляется с округлением по следующему правилу: результат деления двух целых чисел, состоящий из целой и дробной частей, заменяется целой частью, например $20/7=2$, $1/3=0$, $9/3=3$. При таком округлении возникает абсолютная ошибка ε, которая имеет оценку $|\varepsilon|<1$ и имеет знак, противоположный знаку округляемого числа. Заметим, что такой способ округления может давать по сравнению с общепринятым правилом округления в десятичной системе счисления ошибку почти вдвое бóльшую.

Если в вычислениях на ЭВМ присутствует большое число операций деления в арифметике целых чисел, то это может привести к существенному искажению результата — большой погрешности из-за ошибок округления.

В арифметике вещественных чисел на ЭВМ все четыре действия, как правило, выполняются с округлением. Исключение составляют те числа и такие операции, которые не выводят исходные числа и результат операции за рамки 32-разрядного слова ЭВМ (в арифметике одинарной точности) двоичного представления чисел.

Продемонстрируем возникновение ошибки операции сложения на следующем примере. Пусть производится сложение чисел $0{,}1234011$ и $0{,}1234067\cdot 10^{-3}$. При сложении на ЭВМ предварительно выравниваются порядки чисел до большего, а затем мантиссы суммируются. Второе число при выравнивании порядка представится в виде $0{,}0001234067$. Учитывая, что в 32 разрядах точность представления чисел ~ 7 знаков после запятой, замечаем, что подчеркнутые цифры выходят за рамки 32 разрядов, они округляются и сумма $0{,}1235245$ имеет погрешность $\sim 10^{-8}$.

Операция округления осуществляется в двоичном представлении, где точность определяется 24 двоичными цифрами мантиссы. Обозначим точное число — x, округленное — $\tilde{x}$. Тогда относительная ошибка округления δ имеет оценку

$$|\delta| \leqslant 2^{-25} \sim 3\cdot 10^{-8};$$

следовательно,

$$\tilde{x} = x(1+\delta).$$

Получить априорную оценку погрешности округления для сколько-нибудь содержательных вычислений — крайне трудная задача.

Апостериорная оценка погрешности во многих задачах может быть найдена повторением всего процесса вычислений в арифметике двойной точности и сравнением результатов с полученными в одинарной точности.

● 5.7. Особенности серийных вычислений

В этом пункте рассматриваются особенности методов вычислений, связанные с тем, что задача может решаться многократно (серийные вычисления). При многократном использовании одной и той же программы решения задачи требуется обратить пристальное внимание на ее эффективность, в частности время решения одного варианта.

Пусть время выполнения одного варианта равно T, количество повторений вычислений — N, стоимость единицы времени ЭВМ — q. Тогда стоимость всей серии qNT. Если удается снизить время T на ΔT, то экономия стоимости вычислений $qN\Delta T$ при больших N может быть значительной величиной.

В серийных расчетах вводится понятие *предвычислений.* Это вычислительная работа, которая поизводится для подготовки вычислений одного варианта. Если серия большая (N велико), то можно проводить и большие предварительные вычисления. Пусть время предвычислений T_n; тогда общее время вычислений серии

$$T_n + N(T - \Delta T).$$

Пусть также $\Delta T = cT_n$, $c = \text{const}$, т. е. снижение времени вычисления варианта пропорционально T_n. Тогда общее время вычислений серии

$$T_n + N(T - cT_n)$$

и уже при длине серии N, удовлетвряющей неравенству

$$T_n - NcT_n < 0, \quad N > 1/c,$$

целесообразно проводить предвычисления.

Для примера рассмотрим задачу интегрирования уравнения

$$\frac{dy}{dx} = (\sin x)y^2 + f_k(x), \quad y(0) = 0, \quad 0 < x \leqslant a,$$

на интервале $[0, a]$ для серии функций $f_k(x)$, $1 \leqslant k \leqslant N$, методом Эйлера с шагом h. Имеем

$$y_{i+1} = y_i + h\left[(\sin x_i)y_i + f_k(x_i)\right], \quad y_0 = 0,$$
$$x_i = ih, \quad 1 \leqslant i \leqslant [a/h].$$

Если серия велика, то целесообразно составить таблицу функции $\sin x$ в точках x_i заранее, вычисления вариантов уже вести не обращаясь к функции $\sin x$, а только к таблице. Пусть $[a/h] = 1000$. Время

$$T_n = 1000\, T_{\sin},$$

где $T_{\sin}$ — время вычислений $\sin x$ в одной точке. Пусть $T_{в}$ — время выборки числа, взятое из таблицы. Получаем

$$\Delta T = 1000\left[T_{\sin} - T_{в}\right] \simeq T_{n},$$

поскольку $T_{в} \ll T_{\sin}$. При серии $N = 1000$ сокращение времени вычислений $10^{6} T_{\sin}$.

В большинстве случаев смысл предвычислений состоит в составлении таблиц чисел, обращение к которым значительно сокращает вычислительное время варианта. Может оказаться, что даже таблица умножения окажется предпочтительнее арифметической операции в какой-либо конкретной задаче.

Программирование алгоритмов серийных вычислений также обладает рядом особенностей:

во-первых, возможна оптимизация фортран-программы (см. п. 3.4);

во-вторых, написание часто повторяющихся фрагментов программы на машинно-зависимом языке типа макроассемблера;

в-третьих, аппаратная реализация некоторых фрагментов программы.

Последние два указанных приема относятся уже к более высокому уровню владения вычислительной техникой, нежели тот, который принят в данной книге.

Глава 6
ТЕОРИЯ ПРИБЛИЖЕНИЙ

● 6.1. Введение

Если понятие «приближение» трактовать в широком смысле этого слова, то бóльшая часть методов вычислений окажется элементами теории приближений. Однако в этой главе под теорией приближений понимается узкий круг вопросов, связанных с приближением функций одного или нескольких переменных с помощью других функций. Часто этот круг задач называют задачами аппроксимации функций.

Рассмотрим некоторые практические задачи, приводящие к аппроксимации функций.

6.1.1. Ускорение времени вычислений функций на ЭВМ. Предположим, что решение задачи требует многократного вычисления функции $f(x)$ в различных точках интервала $a \leqslant x \leqslant b$. Функция $f(x)$ задана громоздким аналитическим выражением, например

$$f(x)=\cos\left(2\sin\left(\sqrt{x^2+\sum_{k=1}^{100}\cos^2 kx}\right)\right),\quad a\leqslant x\leqslant b.$$

Тогда естественно стремление заменить функцию $f(x)$ в некотором смысле близкой функцией $g(x)$ (аппроксимировать $f(x)$) так, чтобы

$$\|f(x)-g(x)\|<\varepsilon,$$

где ε — точность аппроксимации. При этом вычисление $g(x)$ должно быть значительно более быстрой процедурой, нежели $f(x)$. Возможно, удастся найти полином 4-й степени $P_4(x)=a_0+a_1x+a_2x^2+a_3x^3+a_4x^4$ такой, что

$$\max_{a\leqslant x\leqslant b}|f(x)-P_4(x)|<\varepsilon, \tag{6.1.1}$$

с удовлетворительной точностью ε аппроксимирующий $f(x)$. Тогда вычисление $f(x)$ заменяется вычислением $P_4(x)=g(x)$;

$$f(x)\sim g(x). \tag{6.1.2}$$

Количество арифметических операций при этом сокращается в сотни раз.

6.1.2. Экономия памяти ЭВМ. Предположим, что функция $f(x)$ задается своими значениями в узлах x_i, $1\leqslant i\leqslant n$, интервала $a\leqslant x\leqslant b$:

$$\begin{array}{cccccc} x_i & x_1 & x_2 & \dots & x_n, \\ \hline f(x_i) & f(x_1) & f(x_2) & \dots & f(x_n) \end{array} \tag{6.1.3}$$

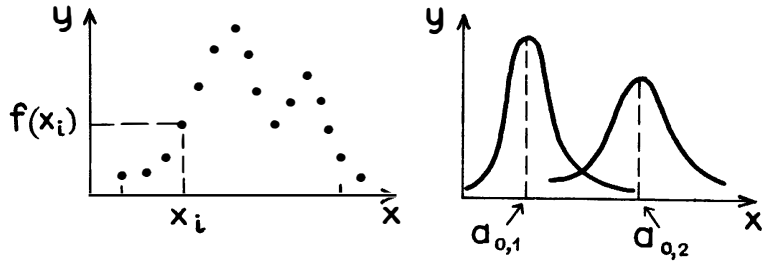

Рис. 6.1

Далее в вычислительном процессе используется эта таблица. При большом числе n хранение таблицы $(x_i, f(x_i))$ в памяти ЭВМ может оказаться слишком обременительным либо вообще невозможным. Тогда естественно возникает задача аппроксимации $f(x)$ близкой функцией $g(x)$, зависящей от небольшого числа параметров, например от пяти $(a_0, a_1, a_2, a_3, a_4)$

$$g(x)=P_4(x),$$

так, чтобы выполнялось условие (6.1.1). При этом в памяти хранится только пять чисел — значения параметров. Значения $f(x)$ теперь не хранятся, а вычисляются приближенно по формуле (6.1.2).

6.1.3. Поиск эмпирических закономерностей по экспериментальным (табличным) данным. Экспериментальные результаты обычно представляются в виде таблиц типа (6.1.3). Экспериментатор на основе практического опыта предполагает, что полученная таблица $f(x)$ является реализацией эмпирического закона $g(x, a)$ с неизвестным параметром a, параметр может быть вектором. Тогда возникает естественная задача определения такого параметра a, при котором эмпирическая закономерность, т. е. $g(x, a)$, наилучшим образом аппроксимировала экспериментальные данные, т. е. $f(x)$.

Например, при исследовании состава вещества на основе спектрального анализа может быть получена таблица функции (6.1.3), изображенная на рис. 6.1. Можно предположить, что спектр $(x_i, f(x_i))$ получается суперпозицией спектров двух веществ. Каждое вещество дает спектр вида (рис. 6.1)

$$g_i(x, a)=a_{2,i}e^{-a_{1,i}(x-a_{0,i})^2}, \quad i=1, 2.$$

Возникает задача поиска параметров $(a_{0,i}, a_{1,i}, a_{2,i})$ таких, что

$$\Big\| \sum_{i=1}^{2} a_{2,i}e^{-a_{1,i}(x-a_{0,i})^2} - f(x_i) \Big\| \to \min. \qquad (6.1.4)$$

Решив эту задачу, можно получить разложение экспериментального спектра на составляющие, т. е. найти оценку вклада каждого компонента в результирующий спектр, а отсюда можно сделать вывод о содержании каждого компонента в веществе.

6.1.4. Классификация задач теории приближений. Простейшая классификация содержит следующие две основные характеристики задачи:

1) норму, в которой осуществляется аппроксимация $\|f(x)-g(x, a)\|$;

2) вид зависимости от параметров a в семействе функций $g(x, a)$.

Мы будем рассматривать только два типа норм:

а) равномерную непрерывную норму

$$\|f\| = \max_{a \leqslant x \leqslant b} |f(x)| ,$$

б) среднеквадратичную интегральную норму

$$\|f\| = \left(\int_a^b f^2(s)\, ds\right)^{1/2}$$

или дискретную норму

$$\|f\| = \left(\frac{1}{m+1} \sum_{j=0}^{m} f^2(x_j)\right)^{1/2},$$

согласованную с интегральной в том смысле, что

$$\left(\frac{1}{(m+1)} \sum_{j=0}^{m} f^2(x_j)\right)^{1/2} \underset{m \to \infty}{\to} \left(\int_0^1 f^2(x)\, dx\right)^{1/2},$$

когда предел существует.

Задачи аппроксимации, использующие норму а) в качестве меры близости, называются *задачами равномерного приближения*. Основополагающие работы в этом направлении принадлежат русскому математику П. Л. Чебышеву (1821—1894).

Задачи аппроксимации, использующие норму б) в качестве меры близости, называются *задачами среднеквадратичного приближения*. Первые работы этого направления связывают с именем немецкого математика К. Гаусса (1777—1855).

По второму признаку задачи теории приближений делятся на линейные и нелинейные. В линейных задачах параметр $a=(a_1, \ldots, a_m)$ входит в семейство $g(x, a)$ линейно, т. е.

$$g(x, a) = \sum_{j=1}^{m} a_j g_j(x),$$

при этом функции $g_j(x)$ нелинейны. Например,

$$g(x, a) = \sum_{j=1}^{m} a_j x^{j-1},$$

$$g(x, a) = \sum_{j=1}^{m} a_j \sin(jx).$$

В нелинейных задачах параметр a входит в семейство нелинейно, например

$$g(x, a)=a_3 e^{-a_2(x-a_1)^2},$$

$$g(x, a)=\sum_{j=1}^{m} \sin(a_j x). \tag{6.1.5}$$

Если в семействе (6.1.5) параметры a_1, a_2 известны, то задача приближения становится линейной.

В этой главе в качестве семейств функций $g(x, a)$ рассматриваются полиномы $P_n(x)$ на всем интервале $[a, b]$ либо функции $S_n(x)$, являющиеся полиномами на подынтервалах $x_i<x<x_{i+1}$ ($x_i \in [a, b]$), так называемые *сплайны*.

Пусть в задаче приближения заданной функции $f(x)$ семейством $g(x, a)$ определяется элемент, реализующий условие

$$\min \| f(x)-g(x, a) \|,$$

тогда этот элемент называется наилучшим.

Например, если

$$g(x, a)=P_n(x)=\sum_{i=0}^{n} a_i x^i$$

и выбирается равномерная норма, то полином, который доставляет

$$\min \max_{a \leqslant x \leqslant b} |f(x)-\sum_{i=0}^{n} a_i x^i|,$$

называют *полиномом наилучшего равномерного приближения*; если выбирается среднеквадратичная норма, то полином, который доставляет

$$\min \left(\frac{1}{m} \sum_{j=1}^{m} (f(x_j)-\sum_{i=0}^{n} a_i x_j^i)^2 \right)^{1/2},$$

называют *полиномом наилучшего среднеквадратичного дискретного приближения*.

● 6.2. Интерполяция

Приближение заданной функции $f(x)$ полиномом n-й степени $P_n(x)$ можно выполнить множеством способов.

Например, если $f(x)$ разлагается в ряд Тейлора на интервале $a \leqslant x \leqslant b$, то отрезок ряда — многочлен Тейлора n-й степени — дает полином, аппроксимирующий $f(x)$:

$$f(x) \simeq f(a)+\frac{f'(a)}{1!}(x-a)+\ldots+\frac{f^{(n)}(a)}{n!}(x-a)^n=P_n(x).$$

Оценка погрешности аппроксимации следует из формулы остаточного члена отрезка ряда Тейлора

$$\max_{a \leqslant x \leqslant b} |f(x)-P_n(x)| \leqslant \frac{(b-a)^{n+1}}{(n+1)!} \max_{a \leqslant x \leqslant b} |f^{(n+1)}(x)|.$$

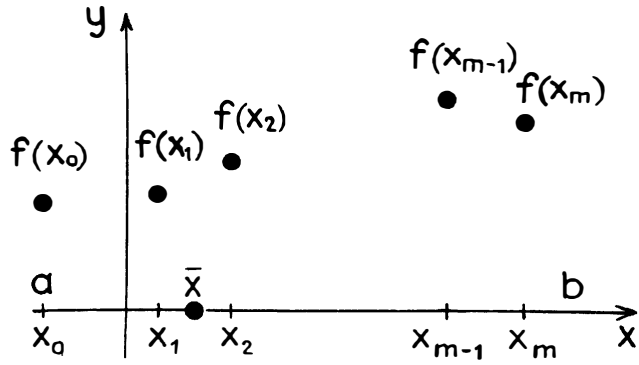

Рис. 6.2

Та же функция может быть представлена многочленом Тейлора с центром в любой точке $c \in [a, b]$, а именно

$$f(x) \simeq f(c) + \frac{f'(c)}{1!}(x-c) + \ldots + \frac{f^{(n)}(c)}{n!}(x-c)^n.$$

Пусть известны значения $f(x)$ в узлах x_j интервала $[a, b]$, $x_0 < x_1 < \ldots < x_m$, $a = x_0$, $b = x_m$, а именно: $f(x_0)$, $f(x_1)$, ... $f(x_m)$. *Задача интерполяции* состоит в том, чтобы найти приближенное значение $f(x)$ в точке $\bar{x} \neq x_j$ (рис. 6.2). Можно предложить метод решения этой задачи, в основе которого лежат ряды Тейлора с центрами в узлах x_j.

Положим

$$f(\bar{x}) \simeq f(x_j) + \frac{f'(x_j)}{1!}(\bar{x}-x_j) + \ldots + \frac{f^{(n)}(x_j)}{n!}(\bar{x}-x_j)^n, \qquad (6.2.1)$$

где x_j — ближайший к $\bar{x}$ узел. Формула (6.2.1) дает пример интерполяционных формул. Основной ее недостаток состоит в том, что используются значения производных функции $f(x)$ в узлах, которые могут быть неизвестны, например, если $f(x)$ задана таблично.

6.2.1. Интерполяционный полином Лагранжа. Свободным от указанного выше недостатка является полином Лагранжа $P_n(x)$, который определяется только значениями $f(x_j)$ из условия

$$P_n(x_j) = f(x_j), \quad 0 \leqslant j \leqslant n. \qquad (6.2.2)$$

Условие (6.2.2) означает, что график полинома n-й степени должен проходить через точки плоскости (x, y) с координатами $(x_j, f(x_j))$ (рис. 6.3).

Покажем, что условие (6.2.2) обеспечивает существование и единственность интерполяционного полинома $P_n(x) = a_0 + a_1 x + \ldots + a_n x^n$.

Теорема 6.1. *Для любых значений $f(x_j)$ и различных узлов x_j, $0 \leqslant j \leqslant n$, существует единственный интерполяционный полином.*

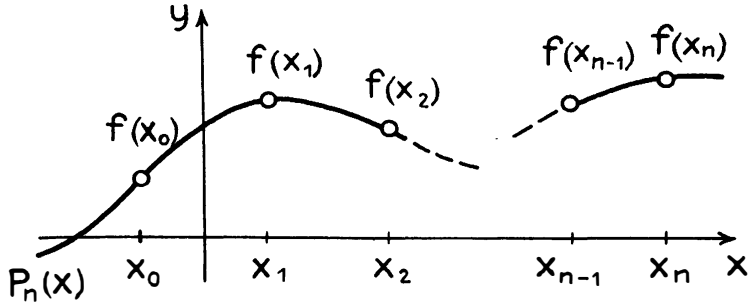

Рис. 6.3

Доказательство. Покажем, что условие (6.2.2) эквивалентно системе линейных алгебраических уравнений относительно коэффициентов полинома $a=(a_0, a_1, \ldots, a_n)$ с определителем, отличным от нуля. Действительно, (6.2.2) в развернутом виде

$$\sum_{i=0}^{n} x_j^i a_i = f(x_j), \quad 0 \leqslant i, \ j \leqslant n,$$

представляет собой систему уравнений относительно вектора a

$$Xa = y, \tag{6.2.3}$$

где матрица X имеет элементы $X_{i,j}=x_j^i$, вектор $y=(f(x_0), f(x_1), \ldots, f(x_n))$. Определитель матрицы X — это определитель Вандермонда

$$\det X = \begin{vmatrix} 1 & x_0 & x_0^2 \ldots x_0^n \\ 1 & x_1 & x_1^2 \ldots x_1^n \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ 1 & x_n & x_n^2 \ldots x_n^n \end{vmatrix} = \prod_{0 \leqslant i \leqslant j \leqslant n} (x_j - x_i) \neq 0,$$

равный произведению П всевозможных разностей узлов. В силу того что узлы различны, $\det X \neq 0$. Поскольку $\det X \neq 0$, решение системы уравнений (6.2.3) существует и единственно, а отсюда следует утверждение теоремы.

Интерполяционный полином Лагранжа можно записать, не решая систему (6.2.3), в виде

$$P_n(x) = f(x_0) \frac{(x-x_1)(x-x_2)\ldots(x-x_n)}{(x_0-x_1)(x_0-x_2)\ldots(x_0-x_n)} +$$

$$+ f(x_1) \frac{(x-x_0)(x-x_2)\ldots(x-x_n)}{(x_1-x_0)(x_1-x_2)\ldots(x_1-x_n)} +$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$+ f(x_n) \frac{(x-x_0)(x-x_1)\ldots(x-x_{n-1})}{(x_n-x_0)(x_n-x_1)\ldots(x_n-x_{n-1})}.$$

Легко видеть, что $P_n(x)$ — полином n-й степени и что выполняется (6.2.2), тогда из теоремы 6.1 следует, что это интерполяционный полином Лагранжа.

Определим полином $(n+1)$-й степени:

$$\omega_{n+1}(x)=(x-x_0)(x-x_1)\ldots(x-x_n).$$

Оценка погрешности интерполяции $f(x)$ с помощью полинома Лагранжа дает следующая теорема.

Теорема 6.2. *Пусть* $f(x)\in C^{(n+1)}[a, b]$, $\bar{x}, x_j\in[a, b]$. *Тогда*

$$f(\bar{x})=P_n(\bar{x})+\frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(\bar{x}), \tag{6.2.4}$$

где точка $\xi=\xi(\bar{x})\in[a, b]$.

Доказательство. Определим функцию

$$u(x)=f(x)-P_n(x)-k\omega_{n+1}(x). \tag{6.2.5}$$

здесь k — не известная пока константа. Заметим, что $u(x)$ имеет на [a, b] по крайней мере $n+1$ нулей, а именно:

$$u(x_j)=0,\ \ 0\leqslant j\leqslant n.$$

Покажем, что число k можно выбрать так, чтобы

$$u(\bar{x})=0,\ \ \bar{x}\neq x_j,\ \ \bar{x}\in(a, b).$$

Для этого следует положить

$$f(\bar{x})-P_n(\bar{x})-k\omega_{n+1}(\bar{x})=0.$$

Поскольку $\omega_{n+1}(\bar{x})\neq 0$, можно найти

$$k=\frac{1}{\omega_{n+1}(\bar{x})}(P_n(\bar{x})-f(\bar{x})).$$

Выразим k через производную $f(x)$. Функция $u(x)$ имеет на $[a, b]$ $n+2$ некратных нулей, функция $u'(x)$ — $n+1$ нулей, ..., $u^{(n+1)}(x)$ — 1 нуль. Обозначим его ξ, значение ξ зависит от точки $\bar{x}$, т. е. $\xi=\xi(\bar{x})$. Имеем

$$u^{(n+1)}(\xi)=0$$

или

$$u^{(n+1)}(\xi)=f^{(n+1)}(\xi)-P_n^{(n+1)}(\xi)-k\omega_{n+1}^{(n+1)}(\xi)=0.$$

Но для любого полинома n-й степени $P_n^{(n+1)}\equiv 0$, для полинома $(n+1)$-й степени $\omega_{n+1}^{(n+1)}=(n+1)!$. Из последнего соотношения находим

$$k=\frac{f^{(n+1)}(\xi)}{(n+1)!}. \tag{6.2.6}$$

Из (6.2.5), (6.2.6) следует утверждение теоремы.

Таким образом, погрешность аппроксимации $f(\bar{x}) \simeq P_n(\bar{x})$ имеет оценку

$$|f(\bar{x}) - P_n(\bar{x})| \leqslant \frac{1}{(n+1)!}\,\omega_{n+1}(\bar{x}) \max_{a \leqslant x \leqslant b} |f^{(n+1)}(x)|.$$

Если нет необходимости находить интерполяционный полином $P_n(x)$, а следует вычислить лишь его значение $P_n(\bar{x})$, то применяется *вычислительная схема Эйткена.* По этой схеме вычисляются последовательно в точке $\bar{x}$ полиномы

$$L_{i,i+1}(\bar{x}) = \frac{1}{x_{i+1} - x_i} \begin{vmatrix} f(x_i) & x_i - \bar{x} \\ f(x_{i+1}) & x_{i+1} - \bar{x} \end{vmatrix};$$

$$L_{i,i+1,i+2}(\bar{x}) = \frac{1}{x_{i+2} - x_i} \begin{vmatrix} L_{i,i+1}(\bar{x}) & x_i - \bar{x} \\ L_{i+1,i+2}(\bar{x}) & x_{i+2} - \bar{x} \end{vmatrix}$$

и т. д. Можно показать, что

$$P_n(\bar{x}) = L_{0,1,2,\ldots,n}(\bar{x}).$$

Вычисления проводятся до тех пор, пока не будет выполняться неравенство

$$|L_{0,1,\ldots,k}(\bar{x}) - L_{0,1,\ldots,k-1}(\bar{x})| < \varepsilon, \quad 2 \leqslant k \leqslant n, \tag{6.2.7}$$

где ε — заданная точность, т. е. в процесс вычислений не вовлекаются лишние узлы, а используется лишь столько узлов, сколько обеспечивает заданную точность ε.

6.2.2. Пример построения интерполяционного полинома Лагранжа. Рассмотрим таблицу

x_j	-1	2	3	5
$f(x_j)$	-1	3	2	4

Следуя формуле (6.2.3), находим

$$P_3(x) = (-1)\frac{(x-2)(x-3)(x-5)}{(-1-2)(-1-3)(-1-5)} + (3)\frac{(x+1)(x-3)(x-5)}{(2+1)(2-3)(2-5)} +$$
$$+ (2)\frac{(x+1)(x-2)(x-5)}{(3+1)(3-2)(3-5)} + (4)\frac{(x+1)(x-2)(x-3)}{(5+1)(5-2)(5-3)}.$$

Можно вычислить приближенное значение $f(x)$ в точке $\bar{x} = 2{,}5$ по формуле

$$f(2{,}5) \simeq P_3(2{,}5).$$

Однако, не зная оценки модуля четвертой производной $|f^{(IV)}(x)|$ на интервале $[-1, 5]$, нельзя оценить погрешность полученного интерполяционного значения. Более того, легко привести пример функции $f(x)$, имеющей те же табличные значения и неограниченной в точке $x = 2{,}5$:

$$f(x) = P_3(x) + \omega_4(x)\frac{1}{(x-2{,}5)}.$$

Этот пример показывает, что, хотя для формального построения интерполяционного полинома достаточно иметь лишь таблицу значений функции $f(x)$, оценка погрешности требует дополнительной информации о поведении $f(x)$ и ее производных на интервале интерполяции. Без оценок соответствующих производных интерполяция практически лишена смысла. Другие формы интерполяционных полиномов можно найти в [2, 4, 24].

6.2.3. Применение программы A3A0. Для вычисления значения интерполяционного полинома Лагранжа в точке $\bar{x}$ по схеме Эйткена можно применять программу A3A0. Пусть функция $f(x)$ задана таблицей

x	1,0	1,1	1,3	1,5	1,6
$f(x)$	1,000	1,049	1,140	1,225	1,265

Требуется вычислить приближенное значение $f(1{,}15)$ с точностью $\varepsilon = 10^{-3}$. Программа может иметь следующий вид:

```
      INTEGER N, I
      REAL X(5),Y(5),Z,E,F(5),P
      DATA X/1., 1.1, 1.3, 1.5, 1.6/
      DATA Y/1., 1.049, 1.14, 1.225, 1.265/
      DATA Z/1.15/, E/0.001/
      N=5
      CALL A3A0(N,X,Y,Z,E,F,P,I)
C     ВЫВОД НА ТЕРМИНАЛ
      WRITE (5,1) I,P
1     FORMAT (2X,'I=',I2,2X,'F=',E10.3)
      END
```

● 6.3. Сплайны

Оценка погрешности интерполяционного полинома указывает на то, что с ростом числа узлов n и соответственно степени полинома $P_n(x)$ может увеличиваться погрешность из-за роста $\|f^{(n)}(x)\|$ (ухудшение гладкости $f(x)$). Кроме того, с ростом n возрастает вычислительная погрешность определения $P_n(\bar{x})$. Эти соображения приводят к другому способу приближения функций — с помощью сплайнов.

6.3.1. Определение сплайна. Пусть интервал $[a, b]$ разбит узлами x_j, как и выше, на n отрезков, $0 \leqslant j \leqslant n$.

Сплайном $S_n(x)$ называется функция, определенная на $[a, b]$, принадлежащая $C^k[a, b]$ и такая, что на каждом интервале $[x_j, x_{j+1}]$, $0 \leqslant j \leqslant n-1$, — это полином n-й степени

$$S_n(x) = a_{0,j} + a_{1,j}x + \ldots + a_{n,j}x^n, \quad x_j \leqslant x \leqslant x_{j+1}.$$

Дефектом сплайна d называется разность между степенью, определяемых его полиномов и порядком гладкости k, т. е.

$$d = n - k.$$

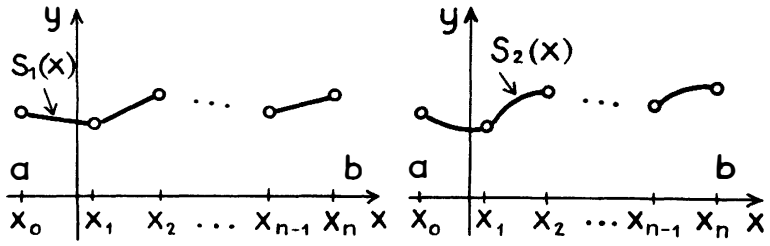

Рис. 6.4

Примеры сплайнов: $S_1(x)$ с дефектом $d=1$, $S_2(x)$ с дефектом $d=2$ — приведены на рис. 6.4. Наиболее распространены сплайны дефекта 1, ниже рассматриваются только такие сплайны.

6.3.2. Параболическая сплайн-интерполяция. Рассмотрим задачу аппроксимации функции $f(x)$, заданной таблично в узлах x_j, $f(x_j)$, сплайном $S_2(x)$ дефекта 1. Определим $S_2(x)$ следующим образом:

$$S_2(x)=S_{2,j}(x)=a_{0,j}+a_{1,j}(x-x_j)+a_{2,j}(x-x_j)^2 \qquad (6.3.1)$$

на каждом интервале $[x_j, x_{j+1}]$. Формула (6.3.1) определяет полином 2-й степени — параболу. Потребуем выполнения во всех узлах условий

$$S_{2,j}(x_j)=f(x_j),\ \ S_{2,j}(x_{j+1})=f(x_{j+1}), \qquad (6.3.2)$$
$$0\leqslant j\leqslant n-1,$$

и дополнительно условий непрерывности первой производной во внутренних узлах, т. е.

$$\frac{d}{dx}S_{2,j}(x_{j+1})=\frac{d}{dx}S_{2,j+1}(x_{j+1}), \qquad (6.3.3)$$
$$0\leqslant j\leqslant n-2.$$

Для определения сплайна $S_2(x)$ по формуле (6.3.1) необходимо найти $3n$ неизвестных:

$$a_{0,j};\ a_{1,j};\ a_{2,j},\ 0\leqslant j\leqslant n-1. \qquad (6.3.4)$$

Условия (6.3.2), (6.3.3) дают $3n-1$ соотношения для определения коэффициентов (6.3.4). Для того чтобы найти коэффициенты, требуется задать еще одно дополнительное условие. Это может быть, например, условие

$$\frac{d}{dx}S_{2,0}(x_0)=\frac{df}{dx}(x_0) \qquad (6.3.5)$$

либо аналогичное условие на правом конце интервала $[a, b]$.

Теорема 6.3. *Условия* (6.3.2), (6.3.3), (6.3.5) *достаточны для определения единственного сплайна* $S_2(x)$.

Доказательство. Покажем, что указанные условия дают возможность определить все коэффициенты (6.3.4). Из (6.3.5) находим

$$a_{1,0}=\frac{df}{dx}(x_0).$$

Из (6.3.2) имеем

$$a_{0,j}=f(x_j),\ \ 0\leqslant j\leqslant n-1,$$

$$a_{0,j}+a_{1,j}(x_{j+1}-x_j)+a_{2,j}(x_{j+1}-x_j)^2=f(x_{j+1}).$$

Отсюда получаем

$$a_{1,j}(x_{j+1}-x_j)+a_{2,j}(x_{j+1}-x_j)^2=f(x_{j+1})-f(x_j). \tag{6.3.6}$$

Из (6.3.3) имеем

$$a_{1,j}+2a_{2,j}(x_{j+1}-x_j)=a_{1,j+1},$$

откуда

$$a_{2,j}=\frac{a_{1,j+1}-a_{1,j}}{2(x_{j+1}-x_j)},\ \ 0\leqslant j\leqslant n-2. \tag{6.3.7}$$

Подставляя (6.3.7) в (6.3.6), получаем формулу

$$a_{1,j+1}=\frac{2(f(x_{j+1}-f(x_j))}{x_{j+1}-x_j}-a_{1,j}, \tag{6.3.8}$$

которая последовательно дает возможность определить все значения $a_{1,1}$, $a_{1,2}$, ..., $a_{1,n-1}$ по известному коэффициенту $a_{1,0}=\dfrac{df}{dx}(x_0)$. Затем по формуле (6.3.7) находим значения $a_{2,0}$, $a_{2,1}$, ..., $a_{2,n-2}$. И, наконец, из (6.3.6) находим

$$a_{2,n-1}=\frac{f(x_n)-f(x_{n-1})}{(x_n-x_{n-1})^2}-\frac{a_{1,n-1}}{(x_n-x_{n-1})}.$$

Таким образом, все коэффициенты (6.3.4) определяются однозначно, что и требовалось доказать.

Построенный сплайн $S_2(x)$ можно теперь использовать для решения задачи интерполяции, а именно определения $f(x)$, $\bar{x}\neq x_j$. Положим $f(\bar{x})\simeq S_2(\bar{x})$. Справедливо следующее утверждение.

Теорема 6.4. *Пусть* $f(x)\in C^3[a, b]$, $x\in(a, b)$. *Тогда*

$$|f(\bar{x})-S_2(\bar{x})|\leqslant ch^3,$$

где $h=\max\limits_{0\leqslant j\leqslant n-1}|x_{j+1}-x_j|$, константа c не зависит от h.

6.3.3. Кубическая сплайн-интерполяция. Интерполяция кубическими сплайнами наиболее популярна в настоящее время. Имеется большое число фортран-программ, реализующих такие алгоритмы [9]. По аналогии с п. 6.3.2 сплайн $S_3(x)$ дефекта 1 задается следующим образом:

$$S_3(x)=S_{3,j}(x)=a_{0,j}+a_{1,j}(x-x_j)+a_{2,j}(x-x_j)^2+a_{3,j}(x-x_j)^3$$

на каждом интервале $[x_j, x_{j+1}]$. Дополнительные условия, определяющие коэффициенты $a_{0,j}$, $a_{1,j}$, $a_{2,j}$, $a_{3,j}$, $0\leqslant j\leqslant n-1$, следующие:

$$S_3(x_j)=f(x_j),\ \ S_3(x_{j+1})=f(x_{j+1}),\ \ 0\leqslant j\leqslant n;$$

во внутренних узлах

$$\frac{d}{dx}S_3(x_j+0)=\frac{d}{dx}S_3(x_j-0),$$

$$\frac{d^2}{dx^2}S_3(x_j+0)=\frac{d^2}{dx^2}S_3(x_j-0).$$

Общее число неизвестных $4n$, число дополнительных условий равно $4n-2$. Недостающие условия задают обычно на краях интервала при $x=x_0$, $x=x_n$. Полученная система линейных уравнений приводится к специальному, так называемому *трехдиагональному* виду. Такие системы эффективно решаются методом прогонки (см. гл. 10). Пусть задаются следующие краевые условия:

$$\frac{d^2S_3}{dx^2}(x_0)=\frac{d^2f}{dx^2}(x_0);\quad \frac{d^2S_3}{dx^2}(x_n)=\frac{d^2f}{dx^2}(x_n).$$

При этом сплайн $S_3(x)$ определяется единственным образом.

Справедливо следующее утверждение.

Теорема 6.5. *Пусть* $f(x)\in C^4[a, b]$, $\bar{x}\in(a, b)$. *Тогда*

$$|f(\bar{x})-S_3(\bar{x})|\leqslant ch^4,$$

где $h=\max\limits_{0\leqslant j\leqslant n-1}|x_{j+1}-x_j|$, *константа* c *не зависит от* h.

Заметим, что главное отличие интерполяции сплайнами от интерполяции полиномами состоит в том, что повышение точности связано не с повышением степени полинома, а с уменьшением расстояния h между узлами. Кроме того, повышение точности не влечет за собой требования существования у $f(x)$ производных все более высокого порядка.

Кубические сплайны дефекта 1 нашли широкое распространение из-за того, что это сплайны минимальной степени, которые в узловых точках имеют непрерывную вторую производную, а следовательно, непрерывную кривизну:

$$k=\frac{\dfrac{d^2S}{dx^2}}{\left[1+\left(\dfrac{dS}{dx}\right)^2\right]^{3/2}}.$$

Непрерывность кривизны является обычным требованием при проектировании кривых и поверхностей различных технических объектов. Проектирование поверхностей связано с двумерной интерполяцией и применением двумерных сплайнов.

6.3.4. Применение программы A3A1. Двумерная поверхность задается набором точек, например, на прямоугольнике D плоскости (x, y) (рис. 6.5). Задача интерполяции, как и в одномерном случае, состоит в приближенном определении $f(x, y)$, где $(x, y)\neq(x_i, y_i)$.

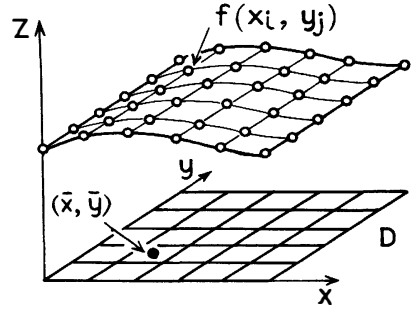

Рис. 6.5

Двумерный сплайн — функция, «сшитая» из кусков двумерных полиномов. Пусть область $D=\{x_0\leqslant x\leqslant x_n;\ y_0\leqslant y\leqslant y_m\}$ разбита на ячейки

$$D_{i,j}=\{x_i\leqslant x\leqslant x_{i+1};\ y_i\leqslant y\leqslant y_{j+1}\},\ 0\leqslant i\leqslant n-1,\ 0\leqslant j\leqslant m-1.$$

Двумерный бикубический сплайн определяется на каждой ячейке $D_{i,j}$ в виде произведения одномерных кубических сплайнов:

$$S(x, y)=A_3(x)B_3(y),$$
$$A_3(x)=a_0+a_1x+a_2x^2+a_3x^3,$$
$$B_3(y)=b_0+b_1y+b_2y^2+b_3y^3.$$

Такое представление позволяет, сопоставляя функции $f(x, y)$, заданной таблицей, интерполяционный бикубический сплайн, конструировать его на основе одномерных сплайнов.

В программе A3A1 реализована интерполяция (приближенное определение $f(\bar{x}, \bar{y})$ по таблице $f(x_i, y_j)$ с помощью одномерной интерполяции с двумя последовательностями шагов.

Первая последовательность:

фиксация y_i, определение $S_1(\bar{x}, y_j)$, $S_{0,1}(\bar{x}, \bar{y})$.

Вторая последовательность:

фиксация x_i, определение $S_0(x_i, \bar{y})$, $S_{1,0}(\bar{x}, \bar{y})$.

Значения $S_{0,1}(\bar{x}, \bar{y})$, $S_{1,0}(\bar{x}, \bar{y})$ можно принять в качестве приближения к $f(\bar{x}, \bar{y})$. Модуль разности $|S_{0,1}(\bar{x}, \bar{y})-S_{1,0}(\bar{x}, \bar{y})$ служит оценкой вычислительной погрешности.

В качестве примера применения программы A3A1 рассмотрим задачу определения f (0,15, 0,25) по таблице $f(x_i, y_j)$:

$$f(x_i, y_j)=(\sin x_i)(\sin y_j),$$
$$x_i=0{,}1\,i,\ 0\leqslant i\leqslant 10,\ y_j=0{,}2j,\ 0\leqslant j\leqslant 5.$$

Программа может иметь следующий вид:

```
      REAL XC,YC,S01,S10,X(11),Y(6),F(11,6),HX,HY,
     * W1(11),W2(11),W3(11),W4(11)
      INTEGER I,J,M,N
      DATA I,J,M,N/0,11,6,11/,XC,YC/0.15,0.25/
      HX=0.1
      HY=0.2
C     ВЫЧИСЛЕНИЕ ЭЛЕМЕНТОВ МАССИВА X,Y,F
      DO 1 I1=1,11
1     X(I1)=HX*(I1-1)
      DO 2 J1=1,6
2     Y(J1)=HY*(J1-1)
      DO 3 I1=1,11
      DO 3 J1=1,6
3     F(I1,J1)=(SIN(X(I1)))*(SIN(Y(J1)))
C     ОБРАЩЕНИЕ К ПРОГРАММЕ A3A1
      CALL A3A1(XC,YC,X,Y,F,SO1,S10,I,W1,
```

```
     *W2,W3,W4,J,M,N)
C      ВЫВОД НА ТЕРМИНАЛ
       WRITE (5,4) S01,S10
4      FORMAT (2X,'S01=',E13.6,2X,'S10=',E13.6)
       END
```

● 6.4. Равномерные приближения

В **6.2** и **6.3** в качестве критерия выбора из класса функций (полиномов или сплайнов) рассматривалось совпадение в узлах x_j функции $f(x)$ и аппроксимирующей функции $g(x, a)$. Получаемая при этом погрешность как в равномерной норме, так и в среднеквадратичной, вообще говоря, может быть уменьшена.

Уменьшить погрешность можно, во-первых, за счет специального распределения узлов интерполяции x_j на интервале $[x_0, x_n]$, если есть возможность выбирать узлы. Ниже будет показано, как следует располагать узлы для интерполяции полиномами.

Во-вторых, можно заменить критерий выбора из класса функций, а именно искать элемент наилучшего приближения, т. е. доставляющий

$$\min_a \| f(x) - g(x, a) \|.$$

В этом пункте рассматривается равномерная норма.

Другими словами, интерполяция состоит в требовании совпадения $f(x)$ и $g(x, a)$ в заданных узлах x_j, погрешность аппроксимации такая — «какая получится»; поиск элемента наилучшего приближения состоит в минимизации погрешности аппроксимации, совпадение $f(x)$ и $g(x, a)$ в некоторых точках такое — «какое получится».

Важную роль в теории равномерного приближения играют полиномы Чебышева. Рассмотрим их основные свойства.

6.4.1. Полиномы Чебышева. Определим полиномы Чебышева исходя из тригонометрических функций $\cos n\theta$, $n=0, 1, 2, \ldots$. Примем в качестве

$$\theta = \arccos x, \quad -1 \leqslant x \leqslant 1.$$

Обозначим

$$T_n(x) = \cos(n \arccos x). \tag{6.4.1}$$

Покажем, что $T_n(x)$ — полином от x, т. е. полином Чебышева. Действительно, по формулам Муавра и бинома Ньютона имеем

$$\cos n\theta + i \sin n\theta = (\cos\theta + i \sin\theta)^n = \sum_{k=0}^{n} C_n^k \cos^{n-k}\theta\, i^k \sin^k\theta.$$

Отсюда, приравняв действительные части, получим

$$\cos n\theta = \sum_{l=0}^{[n/2]} C_n^{2l} \cos^{n-2l}\theta\, i^{2l} \sin^{2l}\theta.$$

Заменим

$$\sin^2\theta=(1-\cos^2\theta).$$

Окончательно имеем

$$\cos n\theta=\sum_{l=0}^{[n/2]}C^{2l}\cos^{n-2l}\theta i^{2l}\,(1-\cos^2\theta)^l. \qquad (6.4.2)$$

Из (6.4.2) следует, что $\cos n\theta$ — полином степени n от $\cos\theta$, но $\cos(\arccos x)=x$. Поэтому

$$T_n(x)=\cos(n\arccos x)=\sum_{l=0}^{[n/2]}C_n^{2l}x^{n-2l}(x^2-1)^l$$

— полином n-й степени от x, $-1\leqslant x\leqslant 1$.

Связь полиномов Чебышева и тригонометрических функций приводит к соотношениям ортогональности полиномов Чебышева на непрерывном интервале и дискретном множестве точек:

$$\int_{-1}^{+1}T_m(x)\,T_n(x)\frac{dx}{\sqrt{1-x^2}}=\int_0^{\pi}\cos m\theta\cos n\theta\,d\theta=\begin{cases}0, & m\neq n,\\ \pi/2, & m=n\neq 0,\\ \pi, & m=n=0;\end{cases} \qquad (6.4.3)$$

$$\sum_{j=0}^{N-1}T_m(x_j)\,T_n(x_j)=\sum_{j=0}^{N-1}\cos m\theta_j\cos n\theta_j=\begin{cases}0, & m\neq n,\\ N/2, & m=n\neq 0,\\ N, & m=n=0,\end{cases} \qquad (6.4.4)$$

здесь $x_j=\cos\theta_j$; $\theta_j=\pi_j/N$, $0\leqslant j\leqslant N-1$.

Из (6.4.3) следует, что полиномы Чебышева ортогональны на интервале $[-1, 1]$ с весовой функцией

$$w(x)=\frac{1}{\sqrt{1-x^2}}.$$

Известна замечательная роль ортогональных систем функций, в частности тригонометрических, для представления функций рядами Фурье.

Ортогональные полиномы, в частности полиномы Чебышева, обладают рядом дополнительных свойств: они удовлетворяют трехчленному рекуррентному соотношению, их легко вычислять и преобразовывать с их помощью степенные ряды.

Трехчленное рекуррентное соотношение

$$T_{n+1}(x)+T_{n-1}(x)=2xT_n(x),\quad n\geqslant 1, \qquad (6.4.5)$$

является следствием тождества

$$\cos(n+1)\theta+\cos(n-1)\theta=2\cos\theta\cos n\theta.$$

Из (6.4.5) можно последовательно определять $T_n(x)$, $n\geqslant 2$, $T_0(x)=1$; $T_1(x)=x$; $T_2(x)=2x^2-1$; $T_3(x)=4x^3-3x$; $T_4(x)=8x^4-8x^2+1$; $T_5(x)=16x^5-20x^3+5x$; $T_6(x)=32x^6-48x^4+18x^2-1$; ...

Графики полиномов $T_0(x)-T_5(x)$ на интервале $[-1, 1]$ приведены на рис. 6.6.

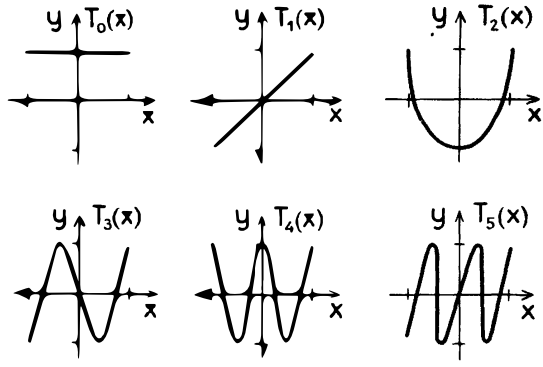

Рис. 6.6

Можно выразить степени переменной x в виде линейных комбинаций полиномов Чебышева

$$x^n=\frac{1}{2^{n-1}}\sum_{k=0}^{[n/2]} C_n^k T_{n-2k}(x). \qquad (6.4.6)$$

Отсюда для первых степеней получаем:

$$1=T_0(x);\ \ x=T_1(x);\ \ x^2=\frac{1}{2}(T_2(x)+T_0(x));$$

$$x^3=\frac{1}{4}(T_3(x)+3T_1(x));\ \ x^4=\frac{1}{8}(T_4(x)+4T_2(x)+3T_0(x));\ldots$$

Используя соотношение (6.4.6), полином

$$P_n(x)=a_0+a_1x+\ldots+a_nx^n$$

можно выразить через полиномы Чебышева (разложить по $T_n(x)$). Например, $P_2(x)=1-2x+3x^2$ имеет вид

$$P_2(x)=T_0-2T_1+\frac{3}{2}(T_2+T_0)=\frac{5}{2}T_0(x)-2T_1(x)+\frac{3}{2}T_2(x).$$

Определим нули и точки экстремумов $T_n(x)$ на интервале $[-1, 1]$. Нули $T_n(x)$ определяются из уравнения

$$\cos(n\arccos x)=0.$$

Отсюда

$$n\arccos x=\pi/2+k\pi,$$

$$x_k=\cos\left(\frac{\pi(2k+1)}{2n}\right),\ \ k=0,\ 1,\ \ldots,\ n-1.$$

Точки экстремума $T_n(x)$ определяются из уравнения

$$\cos(n\arccos x)=\pm 1.$$

Отсюда

$$n\arccos x=\pi m,$$

$$x_m=\cos\frac{\pi m}{n},\ \ m=0,\ 1,\ \ldots,\ n.$$

Итак, $T_n(x)$ на интервале $[-1, 1]$ имеет n вещественных нулей и $n+1$ точку экстремума, при этом экстремальные значения, равные ± 1, чередуются. На оси x эти точки получаются проекцией

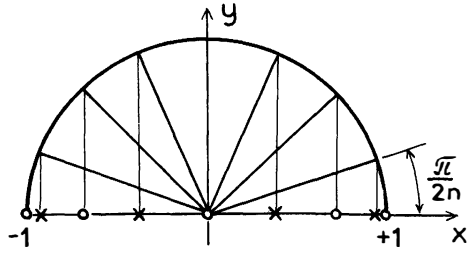

Рис. 6.7

пересечения полукруга с множеством прямых, имеющих между собой равные углы. На рис. 6.7 представлен вариант $n=4$, знаком * обозначены нули $T_4(x)$, знаком 0 — точки экстремума.

Заметим, что старший коэффициент (при x^n) у полинома $T_n(x)$ равен 2^{n-1}.

6.4.2. Теорема Чебышева. *Из всех полиномов $P_n(x)$ n-й степени со старшим коэффициентом, равным единице, у полинома*

$$\bar{T}_n(x)=\frac{1}{2^{n-1}}T_n(x)$$

максимальное абсолютное значение на интервале $[-1, 1]$ наименьшее, т. е.

$$\max_{-1\leqslant x\leqslant 1}|P_n(x)|\geqslant \max_{-1\leqslant x\leqslant 1}|\bar{T}_n(x)|=\frac{1}{2^{n-1}}.$$

Доказательство. Предположим противное:

$$\max_{-1\leqslant x\leqslant 1}|P_n(x)|<\frac{1}{2^{n-1}}. \tag{6.4.7}$$

Рассмотрим полином $(n-1)$-й степени

$$R_{n-1}(x)=P_n(x)-\bar{T}_n(x).$$

В силу (6.4.7) в точках экстремума $\bar{T}_n(x)-x_m$, $m=0, 1, \ldots, n$, полином $R_{n-1}(x_m)$ принимает поочередно разные знаки. Следовательно, между соседними точками x_m полином $R_{n-1}(x)$ имеет по крайней мере один нуль, а их общее число равно n, что невозможно для полинома $(n-1)$-й степени. Противоречие доказывает утверждение теоремы.

Другими словами говорят, что полиномы Чебышева $\bar{T}_n(x)$ — наименее уклоняющиеся от нуля в равномерной норме на интервале $[-1, 1]$.

Теперь очевидным образом решается следующая задача: пусть $f(x)\equiv 0$, $-1\leqslant x\leqslant 1$. Найти полином фиксированной n-й степени

$$g(x, a)=P_n(x)=a_0+a_1x+\ldots+a_{n-1}x^{n-1}+x^n$$

наилучшего равномерного приближения к $f(x)$, т. е. доставляющий

$$\min_{a}\max_{-1\leqslant x\leqslant 1}|P_n(x)|.$$

Теорема Чебышева сразу дает ответ: это полином $\bar{T}_n(x)$.

Для произвольных непрерывных $f(x)$ неизвестны формулы для полинома наилучшего равномерного приближения. Для некоторых классов функций, например рациональных, соответствующие формулы можно найти в [2].

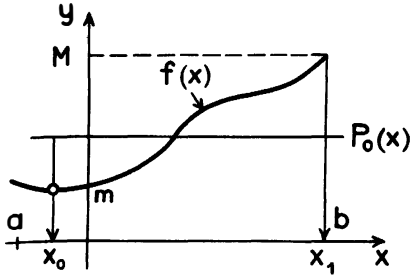

Рис. 6.8

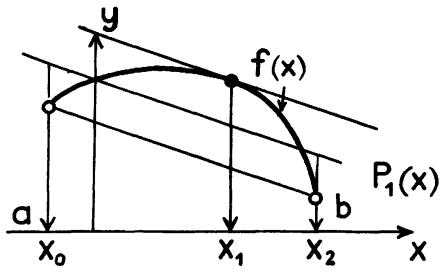

Рис. 6.9

Произвольный интервал $[a, b]$ линейной заменой

$$x_1=\frac{2x-(b+a)}{b-a},\quad a\leqslant x\leqslant b,$$

приводится к интервалу $-1\leqslant x_1\leqslant 1$, где затем и решается задача равномерного приближения.

Важным критерием, устанавливающим свойство полинома наилучшего равномерного приближения, является критерий Чебышева. Сформулируем его без доказательства.

Критерий Чебышева. *Для того чтобы полином $P_n(x)$ был полиномом наилучшего равномерного приближения к непрерывной функции $f(x)$, $a\leqslant x\leqslant b$, необходимо и достаточно существование на $[a, b]$ по крайней мере $n+2$ точек $x_0, x_1, \ldots, x_{n+1}$ таких, что*

$$f(x_i)-P_n(x_i)=\pm(-1)^i\|f(x)-P_n(x)\|. \qquad (6.4.8)$$

Точки x_i, в которых выполняется (6.4.8), называются *точками чебышевского альтернанса.*

Пример 1. Для любого n

$$P_n(x)=a_0+a_1x+\ldots+a_nx^n$$

наименее уклоняющимся от нуля на $[a, b]$ полиномом является $P_n(x)\equiv 0$ (нет ограничения $a_n=1$).

Пример 2. Для $n=0$ полином $P_0(x)=a_0$ наилучшего равномерного приближения к непрерывной $f(x)$ есть

$$P_0(x)=\frac{m+M}{2},\quad m=\min_{a\leqslant x\leqslant b} f(x),\quad M=\max_{a\leqslant x\leqslant b} f(x).$$

Точки чебышевского альтернанса отмечены на рис. 6.8.

Пример 3. Для $n=1$ полином $P_1(x)=a_0+a_1x$ наилучшего приближения к непрерывно дифференцируемой выпуклой на $[a, b]$ функции $f(x)$ строится согласно рис. 6.9. Это прямая, параллельная хорде, соединяющей $f(a)$ и $f(b)$, которая делит пополам расстояние между этой хордой и касательной к $f(x)$, параллельной хорде.

6.4.3. Минимизация оценки погрешности интерполяции за счет выбора узлов. Рассмотрим интерполяцию функции $f(x)$ на интервале $[-1, 1]$. Согласно формуле (6.2.4) имеем

$$f(\bar{x})=P_n(\bar{x})+\frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(\bar{x}),$$

где $\xi=\xi(\bar{x})$, $\bar{x}$ принадлежит интервалу $[-1, 1]$. Отсюда оценка погрешности интерполяции следующая:

$$\max_{-1\leqslant x\leqslant 1}|f(\bar{x})-P_n(\bar{x})|\leqslant\frac{1}{(n+1)!}\max_{-1\leqslant x\leqslant 1}|f^{(n+1)}(x)|\max_{-1\leqslant x\leqslant 1}|\omega_{n+1}(x)|.$$

Заметим, что полином $\omega_{n+1}(x)$ имеет старший коэффициент при x^{n+1}, равный 1. Из теоремы Чебышева вытекает, что

$$\max_{-1\leqslant x\leqslant 1}|\omega_{n+1}(x)|\geqslant\frac{1}{2^n}.$$

Если взять в качестве узлов интерполяции нули полиномов Чебышева $T_{n+1}(x)$

$$x_k=\cos\frac{\pi(2k+1)}{2(n+1)},\quad k=0,\ 1,\ \ldots,\ n,$$

то полином

$$\omega_{n+1}(x)=\bar{T}_{n+1}(x)$$

и оценка погрешности интерполяции будет минимальной, а именно

$$\max_{-1\leqslant x\leqslant 1}|f(\bar{x})-P_n(x)|\leqslant\frac{1}{2^n(n+1)!}\max_{-1\leqslant x\leqslant 1}|f^{(n+1)}(x)|.$$

Таким образом, получаем следующую практическую рекомендацию: *если интерполяция может выполняться с произвольным выбором узлов на интервале* $[-1, 1]$, *то целесообразно в качестве узлов выбрать нули полиномов Чебышева.*

6.4.4. Экономизация степенных разложений. Многие вычисления основаны на использовании степенных разложений функций $f(x)$

$$f(x)=\sum_{k=0}^{\infty}a_kx^k. \tag{6.4.9}$$

При этом бесконечный ряд (6.4.9) заменяется конечной n-й частичной суммой:

$$f(x)\simeq\sum_{k=0}^{n}a_kx^k=S_n(x). \tag{6.4.10}$$

Пусть ряд сходится на интервале $-1\leqslant x\leqslant 1$. Замена (6.4.9) на (6.4.10) приводит к погрешности. Частичная сумма $S_n(x)$ приближает $f(x)$ на интервале $[-1, 1]$ с точностью ε, если

$$\max_{-1\leqslant x\leqslant 1}|f(x)-S_n(x)|\leqslant\delta<\varepsilon.$$

Пусть существуют точки x_* на интервале $[-1, 1]$ такие, что $(n-1)$-я частичная сумма S_{n-1} не приближает $f(x)$ в этих точках с точностью ε, т. е.

$$|f(x_*)-S_{n-1}(x_*)|>\varepsilon.$$

Можно поставить следующий вопрос: есть ли полином $(n-1)$-й степени $P_{n-1}(x)$ такой, чтобы он приближал $f(x)$ с точностью ε, и как его построить? Ответ дает следующее утверждение.

Теорема 6.6. *Пусть выполняется неравенство*

$$\delta+\frac{|a_n|}{2^{n-1}}<\varepsilon. \tag{6.4.11}$$

Тогда существует полином $P_{n-1}(x)$, *аппроксимирующий* $f(x)$ *с точностью* ε:

$$\max_{-1\leqslant x\leqslant 1}|f(x)-P_{n-1}(x)|<\varepsilon, \tag{6.4.12}$$

$P_{n-1}(x)$ *может быть представлен формулой*

$$P_{n-1}(x)=S_{n-1}(x)+a_n\left(x^n-\frac{1}{2^{n-1}}T_n(x)\right). \tag{6.4.13}$$

Доказательство. Представим разность $f(x)-P_{n-1}(x)$ в виде

$$f(x)-P_{n-1}(x)=(f(x)-S_n(x))+(S_n(x)-P_{n-1}(x)).$$

Из (6.4.13) получаем

$$f(x)-P_{n-1}(x)=(f(x)-S_n(x))+\frac{T_n(x)}{2^{n-1}}a_n.$$

Отсюда и из (6.4.11) получаем цепочку неравенств

$$\max_{-1\leqslant x\leqslant 1}|f(x)-P_{n-1}(x)|\leqslant\max_{-1\leqslant x\leqslant 1}|f(x)-S_n(x)|+\frac{|a_n|}{2^{n-1}}\leqslant\delta+\frac{|a_n|}{2^{n-1}}<\varepsilon,$$

т. е. полином (6.4.13) $(n-1)$-й степени удовлетворяет неравенству (6.4.12), что и требовалось доказать.

Процесс перехода от аппроксимации $f(x)$ полиномом n-й степени к аппроксимации полиномом $(n-1)$-й степени с сохранением точности равномерного приближения называется *процессом экономизации степенного разложения*. Этот процесс можно продолжать до тех пор, пока выполняются условия типа (6.4.11).

Для примера найдем полином наименьшей степени, аппроксимирующий $\sin x$, $-1\leqslant x\leqslant 1$, с точностью $\varepsilon=10^{-3}$. Имеем ряд Маклорена

$$\sin x=x-\frac{x^3}{3!}+\frac{x^5}{5!}+\ldots.$$

Погрешность $S_5(x)$ не больше максимума модуля первого отброшенного слагаемого (так как ряд знакопеременный):

$$|\sin x-\left(x-\frac{x^3}{3!}+\frac{x^5}{5!}\right)|\leqslant\frac{1}{7!}<2\cdot 10^{-4}.$$

Проверим условие (6.4.11):

$$2\cdot 10^{-4}+\frac{1}{2^4}-\frac{1}{5!}=7\cdot 2\cdot 10^{-4}<10^{-3}.$$

Находим

$$P_4(x)=x-\frac{x^3}{3!}+\frac{1}{5!}\left(x^5-\frac{1}{2^4}T_5(x)\right)=x-\frac{x^3}{6}+\frac{1}{120}\left(\frac{20}{16}x^3-\frac{5}{16}x\right)=$$

$$=\left(1-\frac{1}{24\cdot 16}\right)x-\left(\frac{1}{6}-\frac{1}{6\cdot 16}\right)x^3.$$

● 6.5. Среднеквадратичные приближения

Здесь рассмотрим приложение широко распространенного метода вычислений — метода наименьших квадратов — в теории приближений.

6.5.1. Среднеквадратичное интегральное приближение. Поставим задачу аппроксимации непрерывной функции $f(x)$ на интервале $\alpha \leqslant x \leqslant \beta$ элементом семейства $g(x, a)$, $a=(a_0, a_1, \ldots, a_n)$:

$$g(x, a)=a_0 g_0(x)+a_1 g_1(x)+\ldots+a_n g_n(x), \tag{6.5.1}$$

где $g_i(x)$ — линейно независимые на $[\alpha, \beta]$ непрерывные функции. При этом выберем $g(x, a)$ таким, чтобы норма отклонения

$$R=\|f-g(x, a)\|=\left(\int_\alpha^\beta (f(x)-g(x, a))^2\,dx\right)^{1/2} \tag{6.5.2}$$

была минимальной, т. е. найдем элемент наилучшего среднеквадратичного интегрального приближения к $f(x)$.

Если $g_i(x)$ — полиномы степени i, например $g_i=\{x^i,\ 0 \leqslant i \leqslant n\}$ или $g_i=\{T_i(x),\ 0 \leqslant i \leqslant n\}$, то это задача поиска полинома наилучшего среднеквадратичного приближения; если $g_i=\{\cos ix,\ 0 \leqslant i \leqslant n\}$, то ищется тригонометрический полином, минимизирующий (6.5.2), и т. д. Основное требование к набору $g_i(x)$ — линейная независимость функций.

Заметим, что при фиксированных функциях $f(x)$, $g_i(x)$, $0 \leqslant i \leqslant n$, норма отклонения R является функцией коэффициентов $(a_0, a_1, \ldots, a_n)$ в линейной комбинации (6.5.1):

$$R=R(a_0, a_1, \ldots, a_n).$$

Выбрать наилучший элемент (6.5.1) — значит найти такие числа $\bar{a}=(\bar{a}_0, \bar{a}_1, \ldots, \bar{a}_n)$, которые доставляют минимум (6.5.2). Следовательно, в точке $\bar{a}$ с необходимостью должны выполняться соотношения

$$\frac{\partial R}{\partial a_0}=0,\ \frac{\partial R}{\partial a_1}=0,\ \ldots,\ \frac{\partial R}{\partial a_n}=0. \tag{6.5.3}$$

Выполняя дифференцирование по a_i в (6.5.2) и подставляя в (6.5.3), получаем систему линейных алгебраических уравнений $(n+1)$-го порядка относительно неизвестных $a_0, a_1, \ldots, a_n$:

$$\begin{aligned}
&\int_\alpha^\beta (f(x)-a_0 g_0(x)-a_1 g_1(x)-\ldots-a_n g_n(x))\,g_0(x)\,dx=0,\\
&\int_\alpha^\beta (f(x)-a_0 g_0(x)-a_1 g_1(x)-\ldots-a_n g_n(x))\,g_1(x)\,dx=0,\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&\int_\alpha^\beta (f(x)-a_0 g_0(x)-a_1 g_1(x)-\ldots-a_n g_n(x))\,g_n(x)\,dx=0.
\end{aligned} \tag{6.5.4}$$

Вводя обозначения для скалярного произведения

$$(u,\ v)=\int_{\alpha}^{\beta} u(x)v(x)\,dx,$$

перепишем систему (6.5.4) в виде

$$\begin{aligned}
&(g_0,\ g_0)a_0+(g_0,\ g_1)a_1+\ldots+(g_0,\ g_n)a_n=(g_0,\ f),\\
&(g_1,\ g_0)a_0+(g_1,\ g_1)a_1+\ldots+(g_1,\ g_n)a_n=(g_1,\ f),\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&(g_n,\ g_0)a_0+(g_n,\ g_1)a_1+\ldots+(g_n,\ g_n)a_n=(g_n,\ f).
\end{aligned}\tag{6.5.5}$$

Теорема 6.7. *Пусть система функций $g_i(x)$ — линейно независима на $[\alpha,\ \beta]$. Тогда для любой непрерывной функции $f(x)$ существует единственный элемент наилучшего среднеквадратичного интегрального приближения вида* (6.5.1).

Доказательство. Покажем, что решение системы (6.5.5) существует и единственно. Действительно, определитель этой системы

$$\begin{vmatrix}
(g_0,\ g_0) & (g_0,\ g_1)\ldots(g_0,\ g_n)\\
(g_1,\ g_0) & (g_1,\ g_1)\ldots(g_1,\ g_n)\\
\ldots\ldots & \ldots\ldots\ldots\ldots\\
(g_n,\ g_0) & (g_n,\ g_1)\ldots(g_n,\ g_n)
\end{vmatrix}$$

является определителем Грама системы функций $g_i(x)$. Можно показать [8], что определитель Грама равен нулю тогда и только тогда, когда система функций $g_i(x)$ линейно зависима. Следовательно, в силу предположения определитель системы (6.5.5) не равен нулю, т. е. решение $\bar{a}$ существует и единственно.

Покажем, что в точке $\bar{a}=(\bar{a}_0,\ \bar{a}_1,\ \ldots,\ \bar{a}_n)$ достигается именно минимум. Заметим, что

$$\begin{aligned}
R^2&=(f-g(x,\ a),\ f-g(x,\ a))=(f-a_0g_0-\ldots-a_ng_n,\ f-a_0g_0-\ldots-a_ng_n)=\\
&=(f,f)+\sum_{i,k=0}^{n}(g_i,\ g_k)a_ia_k-2\sum_{i=0}^{n}(f,\ g_i)a_i.
\end{aligned}\tag{6.5.6}$$

Правая часть в (6.5.6) — квадратичная форма относительно коэффициентов a_i; причем в силу того, что $R\geqslant 0$ для любых a_i, форма в точке экстремума достигает своего неотрицательного минимума; следовательно, и $\sqrt{R^2}$ достигает в точке $\bar{a}$ минимума, что и требовалось доказать.

В качестве примера найдем среднеквадратичную интегральную аппроксимацию функций

$$f(x)=\sqrt{x},\ \ 0\leqslant x\leqslant 1,$$

полиномом первой степени

$$g(x,\ a)=a_0+a_1x,\ \ g_0(x)=1,\ \ g_1(x)=x.$$

Имеем

$$(g_0,\ g_0)=\int_0^1 1^2\,dx=1;\ (g_0,\ g_1)=\int_0^1 x\,dx=\frac{1}{2},$$

$$(g_1,\ g_1)=\int_0^1 x^2dx=\frac{1}{3};\ (g_0,\ f)=\int_0^1 \sqrt{x}\,dx=\frac{2}{3},$$

$$(g_1,\ f)=\int_0^1 x\sqrt{x}\,dx=\frac{2}{5}.$$

Система (6.5.5) принимает вид

$$a_0+\frac{1}{2}a_1=\frac{2}{3},$$

$$\frac{1}{2}a_0+\frac{1}{3}a_1=\frac{2}{5}.$$

Находим: $a_0=4/15,\ a_1=4/5$.

Искомое решение

$$g(x)=\frac{4}{15}+\frac{4}{5}x. \tag{6.5.7}$$

Для этой же функции найдем полином первой степени наилучшего равномерного приближения по схеме примера 3 п. 6.4.2 (рис. 6.10). Точка чебышевского альтернанса x_1 находится из условия $f'(x)=1$. Имеем $1/2\sqrt{x}=1;\ x_1=1/4$. Полином наилучшего равномерного приближения

$$P_1(x)=\frac{1}{8}+x. \tag{6.5.8}$$

Сравнивая (6.5.7), (6.5.8), легко заметить, что полином $g(x)$ минимизирует среднеквадратичное отклонение от $f(x)$, но допускает отдельные ошибки, большие чем $P_1(x)$, например при $x=0$. Наоборот, полином $P_1(x)$ минимизирует наибольшую ошибку, допуская большее среднеквадратичное отклонение, чем $g(x)$.

Наиболее простую форму уравнения (6.5.5) имеют в том случае, когда система функций g_i, $0\leqslant i\leqslant n$, ортогональна, т. е.

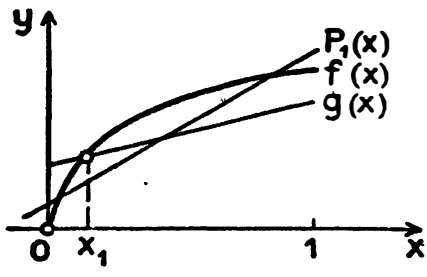

Рис. 6.10

$$(g_i,\ g_k)=\int_a^b g_i(x)\,g_k(x)\,dx=0,\ \ i\neq k.$$

В этом случае коэффициенты a_i записываются из (6.5.5) в явном виде:

$$a_i=\frac{(g_i,f)}{(g_i,\ g_i)},\ \ 0\leqslant i\leqslant n.$$

Поэтому, решая задачу среднеквадратичного приближения, це-

лесообразно выбирать в качестве аппроксимирующих системы ортогональных функций.

Пусть непрерывная функция $f(x)$ задана на интервале $[0, \pi]$. Возьмем в качестве системы функций

$$g_i(x)=\cos ix, \quad 0\leqslant i\leqslant n,$$

$$(g_i,\ g_k)=\int_0^{\pi}\cos ix\cos kxdx=\begin{cases}0, & i\neq k,\\ \pi/2, & i\neq k\neq 0,\\ \pi, & i=k=0.\end{cases}$$

Получаем

$$f(x)\simeq\frac{a_0}{2}+\sum_{i=1}^{n}a_i\cos ix,$$

$$a_i=\frac{2}{\pi}\int_0^{\pi}f(x)\cos ixdx, \quad 0\leqslant i\leqslant n. \tag{6.5.9}$$

Формула (6.5.9) наилучшего среднеквадратичного интегрального приближения по $\cos ix$ оказывается представлением $f(x)$ рядом Фурье на интервале $[0, \pi]$.

Погрешность аппроксимации для ортогональных систем функций g_i получим из формулы (6.5.6), которая принимает следующую форму:

$$R^2=(f,\ f)+\sum_{i=0}^{n}(g_i,\ g_i)a_i^2-2\sum_{i=0}^{n}(g_i,\ g_i)a_i^2=\|f\|^2-\sum_{i=0}^{n}\|g_i\|^2a_i^2.$$

Для ортонормированных систем $(\|g_i\|^2=1)$ формула погрешности упрощается:

$$R^2=\|f\|^2-\sum_{i=0}^{n}a_i^2.$$

Если функция $f(x)$ задана таблично или интегралы в правой части (6.5.5) аналитически не вычисляются, то переходят к среднеквадратичной дискретной аппроксимации.

6.5.2. Среднеквадратичное дискретное приближение. В качестве нормы аппроксимации функции $f(x)$ семейством функций $g(x,\ a)$ на заданном интервале переменного x принимаем

$$R=\left(\frac{1}{(m+1)}\sum_{j=0}^{m}(f(x_j)-g(x_j,\ a))^2\right)^{1/2}, \tag{6.5.10}$$

где x_j — фиксированные узлы интервала. Заметим, что эта норма функций определяется следующим скалярным произведением:

$$(u,\ v)=\frac{1}{m+1}\sum_{j=0}^{m}u(x_j)v(x_j), \quad \|u\|^2=\frac{1}{m+1}\sum_{j=0}^{m}u^2(x_j). \tag{6.5.11}$$

Формула (6.5.10) может быть записана точно так же, как и (6.5.6), а именно

$$R^2=(f(x)-g(x,\ a),\ f(x)-g(x,\ a)).$$

Здесь только нужно помнить, что скалярное произведение задается в m-мерном пространстве векторов $(u(x_0), u(x_1)), \ldots, u(x_m))$ формулой (6.5.11).

Так же, как в п. 6.5.1, поставим задачу аппроксимации функции $f(x)$ элементами семейства $g(x, a)$, $a=(a_0, a_1, \ldots, a_n)$. При этом выберем элемент $g(x, a)$ таким, чтобы норма отклонения от $f(x)$

$$R=\|f-g(x, a)\|,$$

заданная формулой (6.5.10), была минимальной в семействе

$$g(x, a)=a_0g_0(x)+a_1g_1(x)+\ldots+a_ng_n(x),$$

т. е. найдем элемент наилучшего среднеквадратичного дискретного приближения.

Повторяя рассуждения п. 6.5.1, получим, что коэффициенты a_i, $0\leqslant i\leqslant n$, должны удовлетворять системе линейных алгебраических уравнений (6.5.5). Справедлива следующая теорема.

Теорема 6.8. *Пусть $n\leqslant m$, узлы x_j, $0\leqslant j\leqslant m$, таковы, что определитель матрицы*

$$G_{i,k}=(g_i, g_k), \quad 0\leqslant i, k\leqslant n,$$

отличен от нуля. Тогда для любой функции $f(x)$ существует единственный элемент наилучшего среднеквадратичного дискретного приближения в семействе $g(x, a)$.

Коэффициенты a_i определяются решением системы (6.5.5).

При наличии $n=m$ и различных узлов x_j элемент наилучшего приближения является интерполяционным полиномом, в этом случае $R=0$.

Рассмотрим некоторые примеры выбора семейств функций $g(x, a)$.

Пример 1. Семейство $g(x, a)$ — полиномы n-й степени

$$g(x, a)=a_0+a_1x+a_2x^2+\ldots+a_nx^n. \tag{6.5.12}$$

Из (6.5.12) определяем $g_i(x)$, $0\leqslant i\leqslant n$,

$$g_0(x)=1; \ g_1(x)=x; \ g_2(x)=x^2; \ \ldots; \ g_n(x)=x^n.$$

Скалярные произведения

$$(g_i, g_k)=\frac{1}{m+1}\sum_{j=0}^{m} g_i(x_j)g_k(x_j),$$

$$(g_i, f)=\frac{1}{m+1}\sum_{j=0}^{m} g_i(x_j)f(x_j)$$

с учетом того, что $g_i(x)=x^i$, принимают вид

$$G_{i,k}=(g_i, g_k)=\frac{1}{m+1}\sum_{j=0}^{m} x_j^{i+k}, \quad 0\leqslant i, k\leqslant n,$$

$$b_i=(g_i, f)=\frac{1}{m+1}\sum_{j=0}^{m} x_j^i f(x_j), \quad 0\leqslant i\leqslant n.$$

Можно доказать, что если все узлы $x_0, x_1, \ldots, x_m$ различны, $n \leqslant m$, то

$$\det G_{i,k} \neq 0.$$

Систему (6.5.5) представим в форме

$$\sum_{k=0}^{n} G_{i,k} a_k = b_i, \quad 0 \leqslant i \leqslant n. \tag{6.5.13}$$

Решение (6.5.13) — $(a_0, a_1, \ldots, a_n)$ — определяет полином, наилучший в смысле среднеквадратичного дискретного приближения.

Пусть функция $f(x) = \sqrt{x}$ задана таблично:

x_j	0	0,1	0,2	0,4	0,6	0,9	1,0
$f(x_j)$	0,000	0,316	0,447	0,632	0,775	0,949	1,000

Из (6.5.13) найдем полином первой степени:

$$g(x, a) = a_0 + a_1 x.$$

Имеем

$$7a_0 + 3{,}2a_1 = 4{,}119,$$

$$3{,}2a_0 + 2{,}38a_1 = 2{,}693;$$

отсюда $a_0 = 0{,}185$, $a_1 = 0{,}883$, искомый полином

$$P_1(x) = 0{,}185 + 0{,}883x.$$

Следует обратить внимание на тот факт, что при больших n определение полинома наилучшего приближения вида (6.5.12) становится практически невозможным. Это связано с тем, что строки матрицы $G_{i,k}$ оказываются почти линейно зависимыми, а определитель $G_{i,k}$ имеет значение, близкое к нулю.

В качестве иллюстрации этого явления запишем две последние строки матрицы $G_{i,k}$ рассматриваемого примера при $n = 5$. Имеем

$$b_4 = \sum x_j^4 a_0 + \sum x_j^5 a_1 + \sum x_j^6 a_2 + \sum x_j^7 a_3 + \sum x_j^8 a_4 + \sum x_j^9 a_5,$$

$$b_5 = \sum x_j^5 a_0 + \sum x_j^6 a_1 + \sum x_j^7 a_2 + \sum x_j^8 a_3 + \sum x_j^9 a_4 + \sum x_j^{10} a_5.$$

Вычисляя суммы, получим

$$b_4 = 1{,}813a_0 + 1{,}679a_1 + 1{,}582a_2 + 1{,}508a_3 + 1{,}448a_4 + 1{,}397a_5 = 1{,}739,$$

$$b_5 = 1{,}679a_0 + 1{,}582a_1 + 1{,}508a_2 + 1{,}448a_3 + 1{,}397a_4 + 1{,}354a_5 = 1{,}627.$$

Нетрудно проверить, что

$$b_4 = 1{,}06b_5 + 0{,}03\left(\sum_{i=0}^{5} \alpha_i a_i\right),$$

где числа α_i имеют порядок 1, т. е. с точностью до 0,03 эти строки линейно зависимы.

Решение таких систем линейных уравнений приводит к большой вычислительной погрешности. Чтобы избежать этого, при больших n рекомендуется применять другие семейства $g(x, a)$, в частности

разложения по полиномам Чебышева, которые рассматриваются в примерах 2 и 3.

Пример 2. Пусть аппроксимируемая функция $f(x)$ задана на интервале $[-1, 1]$. Пусть имеется возможность выбрать узлы x_j, $0 \leqslant j \leqslant m$, на которых минимизируется среднеквадратичное дискретное отклонение $g(x, a) - f(x)$.

Выберем в качестве узлов x_j те, на которых ортогональны полиномы Чебышева (см. (6.4.4)), а именно:

$$x_j = \cos \theta_j, \quad \theta_j = \frac{\pi j}{m+1}, \quad 0 \leqslant j \leqslant m.$$

В качестве семейства $g(x, a)$ возьмем

$$g(x, a) = \sum_{i=0}^{n} a_i T_i(x), \quad n \leqslant m, \tag{6.5.14}$$

т. е. $g_i(x)$, $0 \leqslant i \leqslant n$,— полиномы Чебышева

$$g_i(x) = T_i(x), \quad 0 \leqslant i \leqslant n.$$

Скалярное произведение, вычисленное с помощью (6.4.4), $(g_i, g_k) = 0$, $i \neq k$, указывает, что система функций $g_i(x)$ ортогональна. Следовательно, как и в интегральном случае, находим коэффициенты a_i элемента наилучшего среднеквадратичного приближения в явном виде:

$$a_i = \frac{(g_i, f)}{(g_i, g_i)}, \quad 0 \leqslant i \leqslant n.$$

Из (6.4.4) следует, что

$$(g_i, g_i) = \begin{cases} 1, & i = 0, \\ 1/2, & i \neq 0. \end{cases}$$

Таким образом, получаем формулы

$$a_0 = \frac{1}{m+1} \sum_{j=0}^{m} f(x_j),$$

$$a_i = \frac{2}{m+1} \sum_{j=0}^{m} T_i(x_j) f(x_j), \quad 1 \leqslant i \leqslant n.$$

Пример 3. Пусть функция $f(x)$ задана на интервале $[-1, 1]$ таблично в произвольных узловых точках x_j, $0 \leqslant j \leqslant m$. В качестве семейства функций $g(x, a)$ возьмем (6.5.14). Теперь система функций $g_i(x)$, вообще говоря, не является ортогональной и необходимо решать систему линейных уравнений (6.5.13).

Однако в отличие от примера 1 при больших значениях n не возникает трудностей с решением этой системы.

6.5.3. Применение программы А3А2. Определение коэффициентов элемента наилучшего среднеквадратичного дискретного приближения в случае примера 3 можно выполнить с помощью программы А3А2. В качестве таблично заданной функции возьмем таблицу

из примера 1. Найдем элемент семейства (в обозначениях описания программы A3A2):

$$P_5(x)=\frac{a_{6,1}}{2}T_0(x)+a_{6,2}T_1(x)+a_{6,3}T_2(x)+\ldots+a_{6,6}T_5(x),$$

или в обозначениях (6.5.14)

$$g(x,\ a)=a_0T_0(x)+a_1T_1(x)+\ldots+a_5T_5(x),$$

минимизирующий $(m=6)$

$$R=\left(\frac{1}{m+1}\sum_{j=0}^{m}(f(x_j)-g(x_j,\ a))^2\right)^{1/2},$$

или (в обозначениях описания программы)

$$\varepsilon_6=\left(\sum_{l=1}^{7}w_l(y_l-P_5(x_l))^2\right)^{1/2}.$$

Заметим, что интервал задания $f(x)$ должен быть $[-1,\ 1]$. Сравнивая $P_5(x)$ и $g(x,\ a)$, R и ε_6, получаем соответствие:

$$a_0=a_{6,1}/2;\ \ a_1=a_{6,2};\ \ a_2=a_{6,3};\ \ \ldots;\ \ a_5=a_{6,6};$$

$$w_l=\frac{1}{7},\ \ l=1,\ 2,\ \ldots,\ 7,\ \ y_l=f(x_{l-1}),$$

$$l=1,\ 2,\ \ldots,\ 7.$$

Кроме того, следует выполнить преобразование x_j к интервалу $[-1,\ 1]$

$\tilde{x}_l=2x_{l-1}-1,\ \ l=1,\ 2,\ \ldots,\ 7.$

Программа может иметь следующий вид:

```
      INTEGER M,K1,N,I
      REAL X(7),Y(7),W(7),W1(3,7),W2(2,6),A(6,6),E(6)
      DATA X/0.,0.1,0.2,0.4,0.6,0.9,1.0/
      DATA Y/0.,0.316,0.447,0.632,0.775,0.949,1.0/
      DATA M,K1,N,I/7,6,6,0/
C     ПРИВЕДЕНИЕ МАССИВА X К СТАНДАРТНОМУ ИН-
C     ТЕРВАЛУ
C     ЗАДАНИЕ ВЕСОВЫХ КОЭФФИЦИЕНТОВ
      DO 1 L=1,7
      X(L)=2.*X(L)-1.
1     W(L)=1./7.
C     ОБРАЩЕНИЕ К ПРОГРАММЕ A3A2
      CALL A3A2(M,K,N,X,Y,W,W1,W2,A,E,I)
C     ВЫВОД НА АЦПУ
      WRITE (6,2) (A(6,I),I=1,6),E(6)
2     FORMAT (2X,6E10.3,2X,E10.3)
      END
```

Глава 7
ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ

● 7.1. Введение

7.1.1. Численное интегрирование. Во многих научных и технических задачах интегрирование функций является составной частью решения полной проблемы. Вычисление площадей и объемов, определение центра и моментов инерции тел, вычисление значения работы, произведенной некоторыми силами, и многие другие задачи приводят к интегрированию функций. Геометрический смысл простейшего определенного интеграла

$$I=\int_a^b f(x)\,dx \tag{7.1.1}$$

от неотрицательной функции $f(x)\geqslant 0$, как известно, состоит в том, что значение I — это площадь, ограниченная кривой $y=f(x)$, осью абсцисс и прямыми $x=a$, $x=b$ (рис. 7.1).

Заметим, что неопределенные интегралы от элементарных функций могут уже не выражаться через элементарные функции, например

$$\int \frac{\sin x}{x}\,dx.$$

Тогда, если нет возможности выразить интеграл в известных специальных функциях, для которых имеются таблицы или программы вычисления на ЭВМ, то применяется приближенное численное интегрирование (7.1.1). Кроме того, если $f(x)$ задана таблично, то приближенное определение интеграла (7.1.1) также выполняется численно.

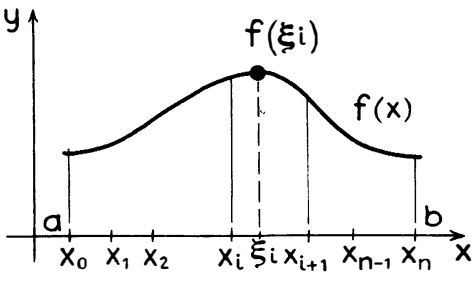

Рис. 7.1

Напомним определение интеграла Римана от $f(x)$, формально записываемого в виде (7.1.1). В этом определении фактически уже заложена основная идея численного интегрирования.

Пусть вещественная функция $f(x)$ определена и ограничена на замкнутом интервале $[a,\ b]$. Разобьем $[a,\ b]$

на n частичных интервалов $[x_i,\ x_{i+1}]$, $0 \leqslant i \leqslant n-1$, $x_n = b$, $x_0 = a$. Выберем в каждом частичном интервале произвольную точку ξ_i, $x_i \leqslant \xi_i \leqslant x_{i+1}$, и составим интегральную сумму (рис. 7.1):

$$S = \sum_{i=0}^{n-1} f(\xi_i)(x_{i+1} - x_i). \tag{7.1.2}$$

Если существует предел S при стремлении длины наибольшего частичного интервала к нулю и произвольных ξ_i, то этот предел называется *интегралом Римана* от $f(x)$

$$I = \lim_{\max_i |x_{i+1} - x_i| \to 0} S. \tag{7.1.3}$$

Интеграл Римана существует (существует предел (7.1.3)), например, для непрерывных функций $f(x) \in C[a,\ b]$.

Вычисление суммы (7.1.2), если не переходить к пределу (7.1.3), дает простейший пример численного интегрирования. А верхняя S_2 и нижняя S_1 суммы Дарбу определяют величину погрешности S, а именно:

$$\begin{gathered} |I - S| \leqslant S_2 - S_1 \\ S_1 = \sum_{i=0}^{n-1} m_i(x_{i+1} - x_i);\ m_i = \min_{x_i \leqslant x \leqslant x_{i+1}} f(x), \\ S_2 = \sum_{i=0}^{n-1} M_i(x_{i+1} - x_i),\ M_i = \max_{x_i \leqslant x \leqslant x_{i+1}} f(x). \end{gathered} \tag{7.1.4}$$

Разнообразные формулы численного интегрирования, по существу, отличаются от (7.1.2) только явным указанием способа:

1) выбора x_i, ξ_i;
2) ускорения сходимости в (7.1.3);
3) оценки погрешности, использующей дополнительную информацию о поведении $f(x)$ (например, информации, что $f(x) \in C^2[a,\ b]$).

Если же об интегрируемой функции $f(x)$ ничего не известно, кроме того, что $f(x)$ непрерывна, то вычисление суммы (7.1.2) и оценка погрешности (7.1.4) являются наиболее естественной формулой численного интегрирования.

7.1.2. Определение квадратурной формулы. Обобщим понятие интегральной суммы (7.1.2).

Точки ξ_i, в которых вычисляются значения $f(x)$, назовем *узлами*, а коэффициенты $(x_{i+1} - x_i)$ в (7.1.2) заменим некоторыми числами q_i, не зависящими от $f(x)$, называемыми *весами*. Тогда формула (7.1.2) заменяется следующей:

$$Q = \sum_{i=0}^{n-1} q_i f(\xi_i), \tag{7.1.5}$$

где $a \leqslant \xi_i \leqslant b$. Запишем интеграл (7.1.1) в виде

$$\int_a^b f(x)\,dx = \sum_{i=0}^{n-1} q_i f(\xi_i) + R. \tag{7.1.6}$$

Формула (7.1.5) называется *квадратурной формулой*, R в (7.1.6) — погрешностью квадратурной формулы.

Каждая конкретная квадратурная формула считается заданной, если указано, как выбирать ξ_i, соответствующие веса q_i, а также оценка погрешности R для определенных классов функций.

7.1.3. Точные квадратурные формулы. Для некоторых классов функций можно записать квадратурные формулы с погрешностью

$$R \equiv 0$$

сразу для всего класса. Такие квадратурные формулы называются *точными*.

Запишем точные квадратурные формулы для полиномов степени m, т. е. для

$$P_m(x) = a_0 + a_1 x + \ldots + a_m x^m$$

на интервале $[a, b]$. Определим на $[a, b]$ произвольные попарно различные узлы ξ_i, $0 \leqslant i \leqslant m$. Найдем веса q_i такие, что

$$\int_a^b P_m(x)\,dx = \sum_{i=0}^{m} q_i P_m(\xi_i). \tag{7.1.7}$$

Перепишем $P_m(x)$ в виде интерполяционного полинома:

$$P_m(x) = \sum_{i=0}^{m} P_m(\xi_i) \frac{(x-\xi_0)\ldots(x-\xi_{i-1})(x-\xi_{i+1})\ldots(x-\xi_m)}{(\xi_i-\xi_0)\ldots(\xi_i-\xi_{i-1})(\xi_i-\xi_{i+1})\ldots(\xi_i-\xi_m)}.$$

Условие (7.1.7) приводит к следующим значениям для весов $0 \leqslant i \leqslant m$:

$$q_i = \int_a^b \frac{(x-\xi_0)\ldots(x-\xi_{i-1})(x-\xi_{i+1})\ldots(x-\xi_m)}{(\xi_i-\xi_0)\ldots(\xi_i-\xi_{i-1})(\xi_i-\xi_{i+1})\ldots(\xi_i-\xi_m)}\,dx. \tag{7.1.8}$$

Итак, если взять произвольные различные узлы ξ_i на $[a, b]$, вычислить значения весов по формулам (7.1.8), то для любого полинома степени m квадратурная формула

$$Q = \sum_{i=0}^{m} q_i P_m(\xi_i)$$

является точной, т. е. справедливо соотношение (7.1.7).

Может оказаться, что формула (7.1.7) точна для полиномов степени, большей чем m. Это произойдет, если специальным образом на интервале $[a, b]$ выбирать узлы ξ_i. Соответствующие квадратурные формулы построены Гауссом (см. **7.5**). Можно так расположить узлы ξ_i, $0 \leqslant i \leqslant m$, что квадратурная формула Гаусса будет точна для полиномов степени $2m+1$.

Очевидно, что применение точных квадратурных формул для интегрирования полиномов не имеет смысла, так как для них легко находится первообразная $\tilde{P}_{m+1}(x)$ и интеграл

$$I=\tilde{P}_{m+1}(b)-\tilde{P}_{m+1}(a)$$

определяется вычислением полинома $P_{m+1}(x)$ в точках a и b.

Практический смысл точных квадратурных формул проявляется при интегрировании таких классов функций $f(x)$, которые могут быть хорошо аппроксимированы полиномами на интервале $[a, b]$. Тогда, применяя такую формулу к $f(x)$, есть надежда получить малую погрешность R в (7.1.6) для рассматриваемого класса функций.

● 7.2. Простейшие квадратурные формулы

Приведем квадратурные формулы для одного интервала $[x_i, x_{i+1}]$, которые затем обобщаются на весь интервал $[a, b]$ в виде составных квадратурных формул (см. **7.3**).

7.2.1. Формула прямоугольников. Пусть рассматривается интервал $[-h/2, h/2]$, где $h>0$ (рис. 7.2). Предположим, что подынтегральная функция $f(x)$ дважды непрерывно дифференцируема на $[-h/2, h/2]$, т. е. $f(x)\in C^2[-h/2, h/2]$. Запишем соотношение (7.1.6) в виде

$$\int_{-h/2}^{h/2} f(x)\,dx=h\cdot f(0)+R, \qquad (7.2.1)$$

где взят один узел $\xi=0$, соответствующий вес $q=h$. Получаемая квадратурная формула

$$Q=hf(0) \qquad (7.2.2)$$

называется *формулой прямоугольников для одного шага.*

Название оправдывается тем, что (7.2.2) является (для $f(0)>0$) формулой для площади прямоугольника с высотой $f(0)$ и основанием h.

Из рис. 7.2 можно заметить, что если интервал h достаточно мал, а функция $f(x)$—гладкая (мы уже потребовали, чтобы $f(x)\in C^2[-h/2, h/2]$, то погрешность R будет стремиться к нулю при $h\to 0$. Точный результат докажем в следующей теореме.

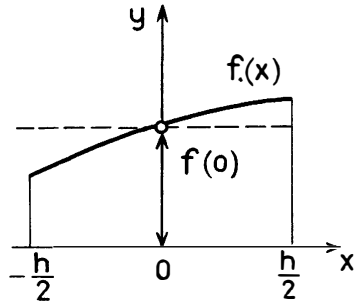

Рис. 7.2

Теорема 7.1. *Пусть* $f(x)\in C^2[-h/2, h/2]$. *Тогда погрешность квадратурной формулы прямоугольников* $R(h, f)$ *имеет вид*

$$R(h, f)=\frac{h^3}{24}f''(\xi), \qquad (7.2.3)$$

где ξ — некоторая точка интервала $[-h/2,\ 2]$.

Доказательство. Перепишем левую часть (7.2.1) в виде

$$\int_{-h/2}^{h/2} f(x)\,dx=\int_{0}^{h/2} f(x)\,dx-\int_{0}^{h/2} f(x)\,dx=F\left(\frac{h}{2}\right)-F\left(-\frac{h}{2}\right). \quad (7.2.4)$$

Здесь функция $F(x)$ определяется формулой $F(x)=\int_0^x f(s)\,ds$. Заметим, что $F(0)=0,\ F'(0)=f(0),\ F''(0)=f'(0),\ F'''(x)=f''(x)$.

По формуле Тейлора с остаточным членом в форме Лагранжа запишем

$$F\left(\frac{h}{2}\right)=F(0)+F'(0)\frac{h}{2}+\frac{1}{2}F''(0)\left(\frac{h}{2}\right)^2+\frac{1}{6}F'''(\xi_1)\left(\frac{h}{2}\right)^3; \quad (7.2.5)$$

$$F\left(-\frac{h}{2}\right)=F(0)-F'(0)\frac{h}{2}+\frac{1}{2}F''(0)\left(\frac{h}{2}\right)^2-\frac{1}{6}F'''(\xi_2)\left(\frac{h}{2}\right)^3, \quad (7.2.6)$$

где $0\leqslant\xi_1\leqslant h/2,\ -h/2\leqslant\xi_2\leqslant 0$. Подставив (7.2.5) и (7.2.6) в (7.2.4), получим

$$\int_{-h/2}^{h/2} f(x)\,dx=hf(0)+\frac{h^3}{24}\left(\frac{f''(\xi_1)+f''(\xi_2)}{2}\right) \quad (7.2.7)$$

Поскольку $f''(x)$ непрерывна на интервале $[-h/2,\ h/2]$, существует точка $\xi\in[-h/2,\ h/2]$ такая, что $\frac{f''(\xi_1)+f''(\xi_2)}{2}=f''(\xi)$. Теперь, сравнивая (7.2.7) и (7.2.1), имеем (7.2.3), что и требовалось доказать.

Квадратурная формула прямоугольников (7.2.2) является точной для полиномов первой степени $P_1(x)=a_0+a_1x$. Действительно,

$$\int_{-h/2}^{h/2}(a_0+a_1x)\,dx=a_0h.$$

Иногда на интервале $[-h/2,\ h/2]$ применяют формулы прямоугольников вида $Q=hf(-h/2)$ или $Q=hf(h/2)$. Нетрудно заметить, что эти формулы точны только для полиномов нулевой степени, т. е. констант.

7.2.2. Формула трапеций. Пусть $[0,\ h]$, $h>0$ (рис. 7.3),— рассматриваемый интервал.

Предположим, что $f(x)\in C^2[0,\ h]$. Запишем соотношение (7.1.6) в виде

$$\int_0^h f(x)\,dx=h\frac{f(0)+f(h)}{2}+R, \quad (7.2.8)$$

где взяты два узла $\xi_0=0,\ \xi_1=h$ и соответствующие веса $q_0=q_1=h/2$. Получаемая квадратурная формула

$$Q=h\frac{f(0)+f(h)}{2} \quad (7.2.9)$$

называется *формулой трапеций для одного шага*. Название связано с тем фактом, что (7.2.9) при положительных $f(0)$, $f(h)$ является формулой для площади трапеции с основаниями $f(0)$, $f(h)$ и высотой h.

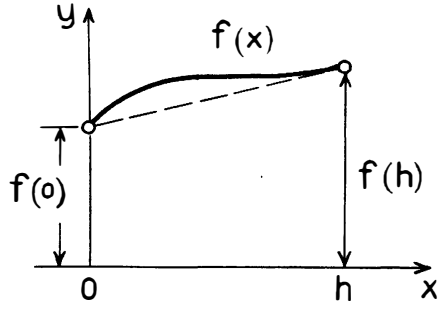

Рис. 7.3

Приведем без доказательства утверждение об оценке погрешности R в (7.2.8).

Теорема 7.2. *Пусть* $f(x)\in C^2[0,\ h]$. *Тогда погрешность квадратурной формулы трапеций* $R(h,\ f)$ *имеет вид*

$$R(h,\ f)=-\frac{h^3}{12}f''(\xi), \tag{7.2.10}$$

где ξ — *некоторая точка интервала* $[0,\ h]$.

Так же как формула прямоугольников, квадратурная формула трапеций (7.2.9) точна для полиномов первой степени.

7.2.3. Формула Симпсона. Пусть $[-h,\ h]$, $h>0$ (рис. 7.4),— рассматриваемый интервал. Предположим, что $f(x)\in C^4[-h,\ h]$.

Запишем соотношение (7.1.6) в виде

$$\int_{-h}^{h} f(x)\,dx=\frac{h}{3}f(-h)+\frac{4h}{3}f(0)+\frac{h}{3}f(h)+R, \tag{7.2.11}$$

где взяты три узла $\xi_0=-h$, $\xi_1=0$, $\xi_2=h$ и соответствующие веса $q_0=q_2=h/3$, $q_1=\frac{4}{3}h$. Получаемая квадратурная формула

$$Q=\frac{h}{3}(f(-h)+4f(0)+f(h)) \tag{7.2.12}$$

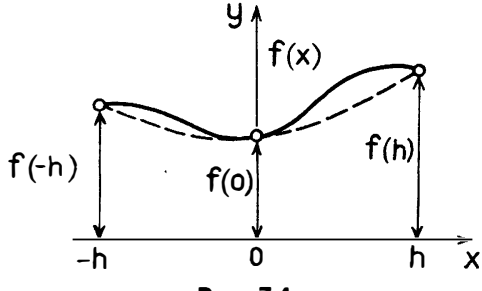

Рис. 7.4

называется *формулой Симпсона* или *формулой парабол*. Последнее название связывается с тем, что (7.2.12) — формула интеграла от параболы, проведенной через точки $(-h,\ f(-h))$, $(0,\ f(0))$, $(h,\ f(h))$ плоскости $(x,\ y)$. Покажем это. Парабола, проходящая через указанные точки, записывается в интерполяционной форме:

$$P_2(x)=f(-h)\frac{x(x-h)}{-h(-h-h)}+f(0)\frac{(x+h)(x-h)}{h(-h)}+f(h)\frac{x(x+h)}{h\cdot 2h}$$

Вычисляем интеграл

$$\int_{-h}^{h} P_2(x)dx=\frac{f(-h)}{2h^2}\left(\frac{x^3}{3}-\frac{x^2}{2}h\right)\Bigg|_{-h}^{h}-\frac{f(0)}{h^2}\left(\frac{x^3}{3}-h^2x\right)\Bigg|_{-h}^{h}+$$

$$+\frac{f(h)}{2h^2}\left(\frac{x^3}{3}-\frac{x^2}{2}h\right)\Bigg|_{-h}^{h}=f(-h)\frac{h}{3}+\frac{f(0)4h}{3}+\frac{f(h)h}{3}$$

и получаем правую часть (7.2.12).

Приведем без доказательства утверждение об оценке погрешности R в (7.2.11).

Теорема 7.3. *Пусть* $f(x)\in C^4[-h, h]$. *Тогда погрешность квадратурной формулы Симпсона* $R(h, f)$ *имеет вид*

$$R(h, f)=-\frac{h^5}{90}f^{(IV)}(\xi), \qquad (7.2.13)$$

где ξ *— некоторая точка интервала* $[-h, h]$.

Из (7.2.13) следует, что квадратурная формула Симпсона точна для полиномов третьей степени.

Применение простейших квадратурных формул требует вычисления значения подынтегральной функции $f(x)$:

в одной точке — для формулы прямоугольников,

в двух точках — для формулы трапеций,

в трех точках — для формулы Симпсона.

Однако, несмотря на малый объем вычислений, область практических приложений простейших формул ограничена лишь малыми интервалами, поскольку при увеличении h погрешность становится значительной. Это следует из формул (7.2.3), (7.2.10) и (7.2.13).

На конечных интервалах интегрирования обычно применяют составные квадратурные формулы.

● 7.3. Составные квадратурные формулы

Если длина интервала $[a, b]$ велика для применения простейших квадратурных формул, то поступают следующим образом:

1) интервал $[a, b]$ разбивают точками x_i, $0\leqslant i\leqslant n$, на n интервалов по некоторому правилу;

2) на каждом частичном интервале $[x_i, x_{i+1}]$ применяют простейшую квадратурную формулу, находят приближенное значение интеграла

$$\int_{x_i}^{x_{i+1}} f(x)dx\cong Q_i, \quad 0\leqslant i\leqslant n;$$

3) из полученных выражений Q_i составляют (отсюда название *составная*) квадратурную формулу для всего интервала $[a, b]$;

4) абсолютную погрешность R составной формулы находят суммированием погрешностей R_i на каждом частичном интервале.

7.3.1. Составные квадратурные формулы с постоянным шагом. Одним из наиболее простых правил разбиения интервала $[a, b]$ на частичные интервалы $[x_i, x_{i+1}]$ является условие

$$x_{i+1}-x_i=h, \quad 0\leqslant i\leqslant n-1, \quad x_0=a, \quad x_n=b.$$

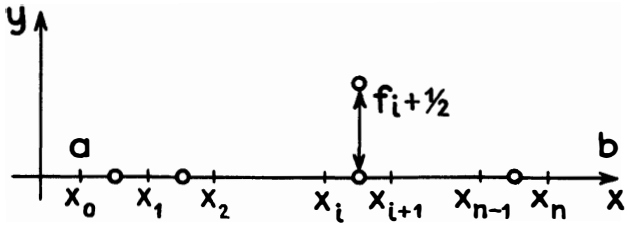

Рис. 7.5

Шаг определяется равенством

$$h=(b-a)/n.$$

Пусть $f(x)\in C^2[a,\ b]$. Обозначим значение функции $f(x)$ в середине интервала $[x_i,\ x_{i+1}]$ (рис. 7.5)

$$f(x_i+h/2)=f_{i+1/2}.$$

Тогда можно переписать (7.2.1) для каждого интервала в виде

$$\int_{x_i}^{x_{i+1}} f(x)dx=hf_{i+\frac{1}{2}}+\frac{h^3}{24}f''(\xi_i), \qquad (7.3.1)$$

$$x_i\leqslant\xi_i\leqslant x_{i+1}.$$

Суммирование по i (7.2.14) приводит к составной формуле прямоугольников с постоянным шагом

$$\int_a^b f(x)dx=h\sum_{i=0}^{n-1} f_{i+\frac{1}{2}}+\frac{h^3}{24}\sum_{i=0}^{n-1} f''(\xi_i).$$

В силу равенства

$$\frac{1}{n}\sum_{i=0}^{n-1} f''(\xi_i)=f''(\xi),\quad a\leqslant\xi\leqslant b,\quad n=\frac{b-a}{h}$$

составная квадратурная формула прямоугольников с погрешностью R принимает окончательный вид

$$\int_a^b f(x)dx=h\sum_{i=0}^{n-1} f_{i+\frac{1}{2}}+R, \qquad (7.3.2)$$

$$R=\frac{h^2(b-a)}{24}f''(\xi).$$

Обозначим (рис. 7.6) значение функции $f(x)$ в точках x_i

$$f(x_i)=f_i.$$

Тогда по аналогии с формулой прямоугольников из (7.2.8) получим составную квадратурную формулу трапеций с постоянным шагом:

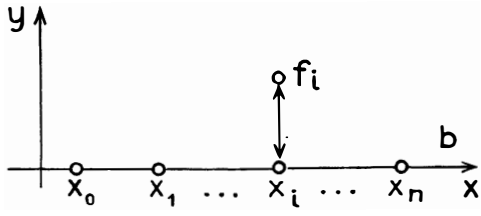

Рис. 7.6

$$\int_a^b f(x)dx=\frac{h}{2}\left(f_0+2\sum_{i=1}^{n-1}f_i+f_n\right)+R,$$

$$R=-\frac{h^2(b-a)}{12}f''(\xi). \tag{7.3.3}$$

Здесь ξ — некоторая точка интервала $[a, b]$.

Для построения составной формулы Симпсона разобьем интервал $[a, b]$ на четное число частичных интервалов $2m$ (рис. 7.7):

$$2m=(b-a)/h.$$

Суммируя (см. (7.2.11))

$$\int_{x_{2i}}^{x_{2i+2}} f(x)dx=\frac{h}{3}(f_{2i}+4f_{2i+1}+f_{2i+2}),\quad 0\leqslant i\leqslant m-1,$$

по i от 0 до $m-1$, получим составную квадратурную формулу Симпсона с постоянным шагом:

$$\int_a^b f(x)dx=\frac{h}{3}\left(f_0+4\sum_{i=1}^{m}f_{2i-1}+2\sum_{i=1}^{m-1}f_{2i}+f_{2m}\right)+R,$$

$$R=-\frac{h^4(b-a)}{180}f^{IV}(\xi). \tag{7.3.4}$$

Здесь ξ — некоторая точка интервала $[a, b]$.

Введем обозначения для квадратурных формул. Формула прямоугольников

$$Q_h^{П}=h\sum_{i=0}^{n-1}f_{i+1/2}. \tag{7.3.5}$$

Формула трапеций

$$Q_h^{Т}=\frac{h}{2}\left(f_0+2\sum_{i=1}^{n-1}f_i+f_n\right). \tag{7.3.6}$$

Формула Симпсона

$$Q_h^{С}=\frac{h}{3}\left(f_0+4\sum_{i=1}^{m-1}f_{2i-1}+2\sum_{i=1}^{m-1}f_{2i}+f_{2m}\right). \tag{7.3.7}$$

Для примера вычислим интеграл

$$I=\int_0^1 e^x dx$$

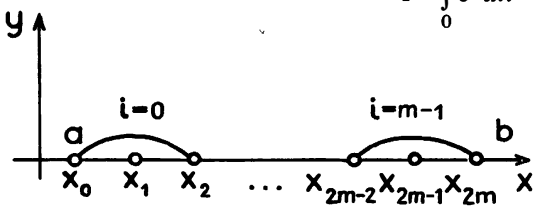

Рис. 7.7

с помощью трех квадратурных формул и сравним ответ с точным значением $J=e-1=$ $=1{,}7182818$. Пусть $h=0{,}1$. Тогда

$$Q_{0,1}^{\text{П}}=0{,}1\left(e^{0,05}+e^{0,15}+e^{0,25}+e^{0,35}+e^{0,45}+e^{0,55}+e^{0,65}+e^{0,75}+\right.$$
$$\left.+e^{0,85}+e^{0,95}\right)=1{,}7176;$$
$$Q_{0,1}^{\text{Т}}=0{,}05\left(e^{0,0}+2\left(e^{0,1}+e^{0,2}+e^{0,3}+e^{0,4}+e^{0,5}+e^{0,6}+e^{0,7}+\right.\right.$$
$$\left.\left.+e^{0,8}+e^{0,9}\right)+e^{1}\right)=1{,}7197;$$
$$Q_{0,1}^{\text{С}}=\frac{0,1}{3}\left(e^{0,0}+4\left(e^{0,1}+e^{0,3}+e^{0,5}+e^{0,7}+e^{0,9}\right)+2\left(e^{0,2}+e^{0,4}+\right.\right.$$
$$\left.\left.+e^{0,6}+e^{0,8}\right)+e^{1}\right)=1{,}7182828.$$

Точное значение I в соответствии с (7.2.15)—(7.2.17) определяет точки ξ в формулах для погрешностей R:

$$I=Q_{0,1}^{\text{П}}+\frac{0,1^2}{24}e^{\xi},\ \xi=0{,}365;$$

$$I=Q_{0,1}^{\text{Т}}-\frac{0,1^2}{12}e^{\xi},\ \xi=0{,}532;$$

$$I=Q_{0,1}^{\text{С}}-\frac{0,1^4}{180}e^{\xi},\ \xi=0{,}588.$$

На практике значение ξ в формулах для R неизвестно, поэтому шаг интегрирования h выбирается по заданной абсолютной точности ε вычисления интеграла из следующих условий:

для формулы прямоугольников

$$\frac{h^2}{24}(b-a)\max_{a\leqslant x\leqslant b}|f''(x)|=\varepsilon;$$

для формулы трапеций

$$\frac{h^2}{12}(b-a)\max_{a\leqslant x\leqslant b}|f''(x)|=\varepsilon;$$

для формулы Симпсона

$$\frac{h^4}{180}(b-a)\max_{a\leqslant x\leqslant b}|f^{\text{IV}}(x)|=\varepsilon.$$

Таким образом, шаг h, а следовательно, число точек n, в которых вычисляется $f(x)$, определяется значениями x с наихудшим поведением $f(x)$ с точки зрения погрешности R.

Такое правило выбора разбиения интервала может приводить к избыточным вычислениям, если $f(x)$ имеет частичные интервалы с «плохим» поведением малой суммарной длины относительно длины $[a,\ b]$ (рис. 7.8).

Примером функции такого типа является $f(x)=e^{-x/\sigma},\ 0\leqslant x\leqslant 1$. Выбирая шаг по наихудшей точке для формулы прямоугольников при $\sigma\ll 1$, имеем значение $h=\sigma\sqrt{24\varepsilon}$, малое для всего интервала $[0,\ 1]$. Почти на всем интервале величина этого шага излишне мелкая для обеспечения заданной точности, а следовательно, будут произведены избыточные вычисления.

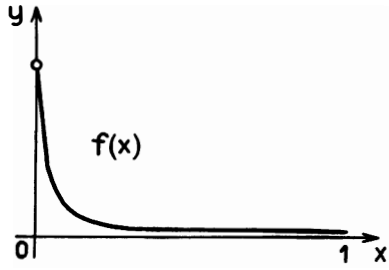

Рис. 7.8

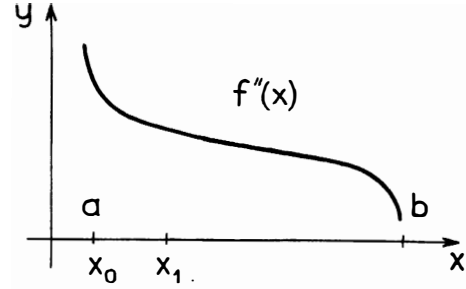

Рис. 7.9

Чтобы освободиться от указанного недостатка составных квадратурных формул с постоянным шагом, исходный интервал разбивают на частичные интервалы различной длины. Причем длина частичных интервалов определяется локальными свойствами $f(x)$ (на данном частичном интервале) и заданной точностью интегрирования.

7.3.2. Составные квадратурные формулы с переменным шагом. Рассмотрим интегрирование функций $f(x)\in C^2[a, b]$ с дополнительным ограничением: $f''(x)$ — монотонная знакоопределенная функция на интервале $[a, b]$.

Пусть для определенности $f''(x)$ — монотонно убывающая положительная функция (рис. 7.9). Положим $x_0=a$. Определим наибольшее значение x_1 из условия, чтобы погрешность простейшей квадратурной формулы прямоугольников R_0

$$\int_{x_0}^{x_1} f(x)\,dx=(x_1-x_0)f\left(\frac{x_0+x_1}{2}\right)+R_0,$$

$$R_0=\frac{(x_1-x_0)^3}{24}f''(\xi),\quad x_0\leqslant\xi\leqslant x_1, \tag{7.3.8}$$

не превышала заданной величины ε. Очевидно, что для этого достаточно решить относительно x_1 уравнение

$$\frac{(x_1-x_0)^3}{24}f''(x_0)=\varepsilon.$$

Имеем

$$x_1=\left(\frac{24\varepsilon}{f''(x_0)}\right)^{1/3}+x_0.$$

Следующие границы частичных интервалов определяются аналогично. Длина частичных интервалов монотонно возрастает. Общая формула такова:

$$x_{i+1}=\left(\frac{24\varepsilon}{f''(x_i)}\right)^{1/3}+x_i,\quad 0\leqslant i\leqslant k. \tag{7.3.9}$$

Количество интервалов k заранее не известно, оно определяется как точностью ε, так и поведением $f''(x)$ на интервале $[a, b]$.

Однако верхняя оценка для k легко определяется по длине наименьшего частичного интервала:

$$k \leqslant \left[\frac{(b-a)(f''(x_0))^{1/3}}{(24\varepsilon)^{1/3}} \right].$$

Суммируя (7.3.8), получим составную квадратурную формулу прямоугольников с переменным шагом:

$$\int_a^b f(x)\,dx = (24\varepsilon)^{1/3} \sum_{i=0}^{k} \frac{f\left(x_i + \frac{1}{2}\left(\frac{24\varepsilon}{f''(x_i)}\right)\right)^{1/3}}{(f''(x_i))^{1/3}} + R, \qquad (7.3.10)$$

где x_i определяется рекуррентно формулами (7.3.9). Для погрешности R имеем оценку $|R| \leqslant k\varepsilon$.

Если $f''(x)$ — монотонно возрастающая положительная функция, то частичные интервалы определяются справа налево от точки b к a. Для отрицательной функции $f''(x)$ и монотонно возрастающей частичные интервалы определяются слева направо от a к b, для убывающей — справа налево от b к a.

В качестве иллюстративного примера рассмотрим задачу интегрирования на интервале $[0, 1]$ функции

$$f(x) = e^{-x/\sigma}, \quad \sigma = 10^{-2}$$

с точностью $\varepsilon = 10^{-4}$ на каждом частичном интервале. Определяя границы интервалов по формуле (7.3.9), получаем: $x_0 = 0{,}000$; $x_1 = 0{,}00621$; $x_2 = 0{,}0138$; $x_3 = 0{,}0237$; $x_4 = 0{,}0374$; $x_5 = 0{,}0590$; $x_6 = 0{,}103$, $x_7 = 0{,}299$; $x_8 = 1{,}000$. Таким образом, общая погрешность составной квадратурной формулы прямоугольников с переменным шагом, полученная на восьми частичных интервалах, имеет оценку $R \leqslant 8 \times \times 10^{-4}$. Такую же погрешность с помощью составной квадратурной формулы прямоугольников с постоянным шагом можно получить на 721-м частичном интервале

$$h = \frac{1}{\sigma\sqrt{24R}} \simeq 721,$$

если выбирать шаг h на всем интервале из условия

$$\frac{h^2}{24}(1-0) \max_{0 \leqslant x \leqslant 1} (e^{-x/\sigma})'' = R.$$

Если $f''(x)$ на всем интервале $[a, b]$ не удовлетворяет принятому дополнительному ограничению, то следует сначала разбить интервал $[a, b]$ на частичные интервалы, на которых $f''(x)$ монотонна и знакоопределенна, а затем на каждом из них построить составную квадратурную формулу с переменным шагом (рис. 7.10) по приведенным выше формулам.

Квадратурные формулы (7.2.19), (7.20.20) были построены на основе простейшей формулы прямоугольников. Аналогичные

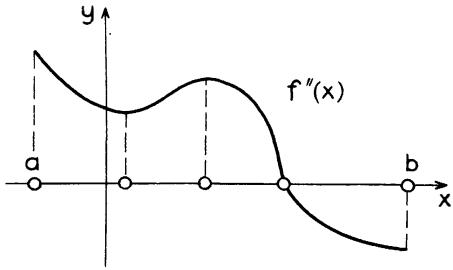

Рис. 7.10

формулы можно вывести и для простейшей формулы Симпсона с той лишь разницей, что нужно интервал $[a, b]$ разбить на частичные интервалы, на которых $f^{IV}(x)$ монотонна и знакоопределенна. Более общие методы интегрирования с переменным шагом изложены в [2].

Наряду с достоинствами квадратурных формул с переменным шагом, проиллюстрированными на примере, отметим их недостатки.

1. В формуле (7.3.9) или (7.3.10) необходимо вычислять значение $f''(x)$ — это дополнительная работа по сравнению с (7.3.2).

2. Получение информации о поведении $f''(x)$ — определение интервалов монотонности и знакоопределенности — это также дополнительная вычислительная работа.

Поэтому, прежде чем отказываться от постоянного шага интегрирования и переходить на переменный, следует убедиться, что достоинства превосходят недостатки. Как правило, дополнительная вычислительная работа оправдывается при серийных вычислениях (см. гл. 5).

● 7.4. Оценка погрешности численного интегрирования

7.4.1. Общая погрешность численного интегрирования. Формула численного интегрирования имеет вид

$$I=\int_a^b f(x)\,dx=Q+R, \tag{7.4.1}$$

где

$$Q=\sum_{i=0}^{n-1} q_i f(\xi_i) \tag{7.4.2}$$

— квадратурная формула, R — погрешность квадратурной формулы. При вычислении по формуле (7.4.2) вносится погрешность из-за погрешности задания значений весов и погрешности вычисления $f(\xi_i)$, связанной с погрешностью задания узлов.

Таким образом, абсолютная погрешность численного определения интеграла по формуле (7.4.2) может быть представлена (см. гл. 5) в виде

$$\Delta I=\sum_{i=0}^{n-1}\left(\Delta q_i \max|f(\xi_0)|+q_i \max\left|\frac{df(\xi_i)}{dx}\right|\Delta\xi_i\right)+|R|, \tag{7.4.3}$$

где Δq_i — абсолютная погрешность весов, $\Delta\xi_i$ — абсолютная погрешность узлов.

При вычислении на ЭВМ можно считать

$$\Delta q_i \cong \Delta \xi_i \simeq \varepsilon_b, \quad 0 \leqslant i \leqslant n-1,$$

где ε_b — точность представления вещественных чисел (в одинарной точности $\varepsilon_b \simeq 10^{-7}$, в двойной $\varepsilon_b \simeq 10^{-17}$). Всюду в **7.4** будем считать, что функции $f(x)$ по крайней мере дважды непрерывно дифференцируемы. Однако уже из формулы (7.4.3) следует, что для оценки погрешности ΔI важно иметь числовую оценку модуля функции и ее производных.

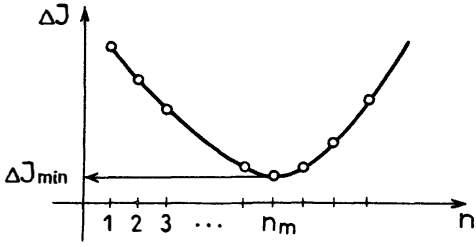

Рис. 7.11

Предположим, что

$$\max_{a \leqslant x \leqslant b} |f(x)| \leqslant M_0, \quad \max_{a \leqslant x \leqslant b} \left| \frac{df}{dx}(x) \right| \leqslant M_1. \tag{7.4.4}$$

Учитывая оценки (7.4.4), перепишем (7.4.3):

$$\Delta I = n\varepsilon_b (M_0 + M_1) + |R|.$$

Предположим, что

$$\max_{a \leqslant x \leqslant b} \left| \frac{d^2 f}{dx^2}(x) \right| \leqslant M_2; \quad \max_{a \leqslant x \leqslant b} \left| \frac{d^4 f}{dx^4}(x) \right| \leqslant M_4.$$

Тогда из (7.3.2)—(7.3.4) получим выражения для ΔI формул прямоугольников, трапеций, Симпсона:

$$\begin{aligned} \Delta I^{\text{П}} &= n\varepsilon_b (M_0 + M_1) + \frac{(b-a)^3 M_2}{24n^2}, \\ \Delta I^{\text{Т}} &= n\varepsilon_b (M_0 + M_1) + \frac{(b-a)^3 M_2}{12n^2}, \\ \Delta I^{\text{С}} &= n\varepsilon_b (M_0 + M_1) + \frac{(b-a)^5 M_4}{180n^4}. \end{aligned} \tag{7.4.5}$$

Из (7.4.5) можно заключить, что существует абсолютная погрешность $\Delta I_{\min}$ для любой квадратурной формулы, которую нельзя уменьшить увеличивая число шагов n (рис. 7.11). Конкретное значение $\Delta I_{\min}$ зависит от величин M_0, M_1, M_2, M_4, a, b, ε_b.

Например, для формулы Симпсона получаем:

$$\frac{d\Delta I^{\text{С}}}{dn} = \varepsilon_b (M_0 - M_1) - \frac{(b-a)^5 M_4}{45n^5} = 0;$$

$$n_m = (b-a) \left(\frac{M_4}{45\varepsilon_b (M_0 + M_1)} \right)^{1/5};$$

$$\Delta I_{\min} = \varepsilon_b^{4/5} (M_0 + M_1)^{4/5} M_4^{1/5} \left(\frac{1}{45^{1/5}} + \frac{(b-a)45^{4/5}}{180} \right).$$

Пусть $(b-a)=1$, $M_0+M_1=1$, $M_4=45$, $\varepsilon_b=10^{-7}$, тогда $n_m\simeq 25$, $\Delta I_{\min}\simeq 3\cdot 10^{-6}$.

Далее рассматривается только погрешность квадратурной формулы R, однако всегда следует иметь в виду наличие второй компоненты в погрешности, которая ограничивает число разбиений интервала интегрирования на частичные интервалы. Для проверки влияния этой компоненты погрешности переходят к вычислению с двойной точностью, если определение констант M_i оказывается более трудоемкой задачей.

7.4.2. Правило Рунге оценки погрешности квадратурной формулы. Приведенные выше формулы для погрешности квадратурных формул $R(h, f)$ выражаются через значения производных $f(x)$ в некоторой точке $\xi\in[a, b]$, которая практически неизвестна. Получение оценок констант M_i требует дополнительных вычислений и особенно для таблично заданных функций. В связи с этим получило распространение практическое правило оценки погрешности Рунге, суть которого состоит в том, чтобы, организовав вычисления двух значений интеграла по двум семействам узлов, затем сравнить результаты вычислений и получить оценку погрешности. Наиболее популярное правило связано с вычислением интеграла дважды: по шагу h и $h/2$ (рис. 7.12).

Чтобы вывести правило Рунге, предположим, что функция $f(x)\in C^4[a, b]$ для квадратурных формул прямоугольников и трапеций, $f(x)\in C^6[a, b]$ — для формулы Симпсона. Тогда можно показать, что погрешности $R(h, f)$ имеют следующее представление при $h\to 0$:

$$R^{\text{П}}=\left(\frac{1}{24}\int_a^b f''(x)\,dx\right)h^2+O(h^4),$$
$$R^{\text{Т}}=\left(-\frac{1}{12}\int_a^b f''(x)\,dx\right)h^2+O(h^4), \qquad (7.4.6)$$
$$R^{\text{С}}=\left(-\frac{1}{180}\int_a^b f^{\text{IV}}(x)\,dx\right)h^4+O(h^6).$$

Объединив формулы (7.4.6) и (7.3.5)—(7.3.7), запишем точное значение интеграла I по любой из трех квадратурных формул в виде

$$I=Q_h+ch^k+O(h^{k+m}),\quad h\to 0. \qquad (7.4.7)$$

Здесь c — константа в главном члене погрешности; $k=2$, $m=2$ для прямоугольников и трапеций, $k=4$, $m=2$ для формулы Симпсона.

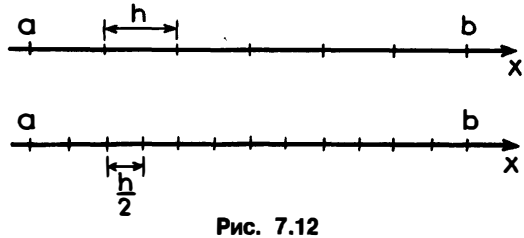

Рис. 7.12

Теорема 7.4. *Пусть подынтегральная функция $f(x)$ такова, что $c\neq 0$. Тогда имеет место соотношение*

$$I = Q_{h/2} + \frac{Q_{h/2} - Q_h}{2^k - 1} + O\left(h^{k+m}\right), \quad h \to 0. \tag{7.4.8}$$

Доказательство. Перепишем (7.4.7) для шага $h/2$, имеем

$$I = Q_{h/2} + c\frac{h^k}{2^k} + O\left(h^{k+m}\right), \quad h \to 0. \tag{7.4.9}$$

Последнее слагаемое записано в том же виде, что и в (7.4.7) в соотвествии с понятием символа O. Из (7.4.7), (7.4.9) определим неизвестную константу c. Вычитая из (7.4.7) равенство (7.4.9), получим

$$0 = Q_h - Q_{h/2} + c\left(h^k - \frac{h^k}{2^k}\right) + O\left(h^{k+m}\right).$$

Отсюда

$$c = \frac{(Q_{h/2} - Q_h)2^k}{h^k(2^k - 1)} + O\left(h^{k+m}\right).$$

Подставляя выражение для c в (7.4.9), получаем (7.4.8), что и требовалось доказать.

В основе правила Рунге лежит соотношение (7.4.8). Для оценки погрешности значения любой квадратурной формулы (7.3.5)—(7.3.7) $Q_{h/2}$ с шагом $h/2$ следует вычислить Q_h с шагом h; тогда погрешность $R_{h/2}$ имеет величину

$$|R_{h/2}| = \left|\frac{Q_{h/2} - Q_h}{2^k - 1}\right| + O\left(h^{k+m}\right),$$

которая на практике заменяется

$$|R_{h/2}| \simeq \frac{|Q_{h/2} - Q_h|}{2^k - 1}$$

с точностью до величин порядка h^{k+m}, т. е. членов более высокого порядка, нежели главный член погрешности в (7.4.6).

Для формул прямоугольников и трапеций

$$|R_{h/2}| \simeq \frac{|Q_{h/2} - Q_h|}{3},$$

для формулы Симпсона

$$|R_{h/2}| \simeq \frac{|Q_{h/2} - Q_h|}{15}.$$

Заметим, что формула прямоугольников неудобна для применения правила Рунге, так как узлы ξ_i формул с $h/2$ и h не совпадают и приходится дополнительно вычислять значения $f(x)$ (по сравнению с формулами трапеций и Симпсона).

Вычисления двух значений квадратурной формулы $Q_{h/2}$ и Q_h позволяют оценить главный член погрешности и уточнить прибли-

женное значение интеграла. В основе уточнения лежит экстраполяция Ричардсона, которая применяется не только к квадратурным формулам, но и к другим задачам численного анализа [16].

Идея экстраполяции Ричардсона состоит в том, чтобы из значений $Q_{h/2}$ и Q_h составить такую линейную комбинацию

$$Q_{h/2,h}=c_1Q_{h/2}+c_2Q_h, \tag{7.4.10}$$

чтобы погрешность приближения I с помощью $Q_{h/2,h}$ была более высокого порядка по h, нежели $Q_{h/2}$ и Q_h в отдельности.

Теорема 7.5. *Пусть точное значение интеграла I представляется в виде* (7.4.7). *Тогда линейная комбинация* (7.4.10) *с коэффициентами*

$$c_1=\frac{2^k}{2^k-1}; \quad c_2=\frac{-1}{2^k-1}$$

имеет погрешность приближения интеграла $O(h^{k+m})$, а именно

$$I=Q_{h/2,h}+O(h^{k+m}).$$

Доказательство следует из прямого сравнения соотношений (7.4.8) и (7.4.10).

7.4.3. Применение программы A4A0. Изложенный выше подход к вычислению интеграла с удвоением числа частичных интервалов и уточнением реализован в программе A4A0; вычисления проводятся по формуле трапеций.

Пусть необходимо вычислить интеграл

$$\int_3^4 \exp(\sin x^2)\,dx$$

с абсолютной точностью $\varepsilon=10^{-4}$. Программа может иметь следующий вид:

```
      INTEGER N,I
      REAL A,B,E,Y,W(10)
      EXTERNAL F
      DATA A,B,E,/3.,4.,1.E—4/,N/10/
C     ОБРАЩЕНИЕ К ПРОГРАММЕ A4A0
      CALL A4A0(A,B,E,N,F,Y,I,W)
C     ВЫВОД НА ТЕРМИНАЛ
      WRITE (5,1) Y,I
1     FORMAT (2X,'Y=',E13.6,'I=',I2)
      END
C     ВНЕШНЯЯ ФУНКЦИЯ—ПОДЫНТЕГРАЛЬНАЯ
C     ФУНКЦИЯ
      FUNCTION F(X)
      F=EXP(—SIN(X*X))
      RETURN
      END
```

● 7.5. Формулы Гаусса

Квадратурные формулы прямоугольников, трапеций и Симпсона, рассмотренные выше, применяются для интегрирования функций $f(x)$ невысокой степени гладкости, для первых двух $f(x)\in C^2[a, b]$. Причем в составных формулах обнаруживается следующий общий дефект: при увеличении числа узлов, в которых вычисляется $f(x)$, для функций высокой степени гладкости ($f(x)\in C^k[a, b]$, $k>2$) не повышается точность с ростом k. Например, для погрешности формулы трапеций из (7.3.3) можно получить оценку снизу

$$|I-Q_h^T|\geqslant\frac{(b-a)}{12}\min_{a\leqslant x\leqslant b}|f''(x)|h^2,$$

которая показывает, что при уменьшении шага h (увеличение числа узлов) и бесконечно дифференцируемых функций погрешность все-таки определяется второй производной $f(x)$.

Указанный дефект называется *явлением насыщения численного метода* и встречается не только в задачах численного интегрирования.

Формулы Гаусса, которые будут ниже определены, являются: 1) квадратурными формулами без насыщения; 2) точными на полиномах максимальной степени $2m+1$ с узлами ξ_i, $0\leqslant i\leqslant m$.

Следует отметить, что достоинства формул Гаусса проявляются только при интегрировании функций достаточно высокой степени гладкости.

Для определения формул Гаусса необходимо ввести понятие полинома Лежандра степени m.

7.5.1. Полиномы Лежандра. В гл. 6 было показано, что полином $P_m(x)$ со старшим коэффициентом, равным единице, наименее уклоняющимся от нуля на интервале $[-1, 1]$ в равномерной норме, является полиномом Чебышева.

Определим полином $P_m(x)$ со старшим коэффициентом, равным единице, наименее уклоняющийся от нуля на интервале $[-1, 1]$ в среднеквадратичной норме. Это и будет полином Лежандра $l_m(x)$.

Для определения коэффициентов $l_m(x)$

$$l_m(x)=a_0+a_1x+\ldots+a_{m-1}x^{m-1}+x^m$$

чисел $(a_0, a_1, \ldots, a_{m-1})$ требуется найти минимум

$$S=\int_{-1}^{1}[a_0,+a_1x+\ldots+a_{m-1}x^{m-1}+x^m]^2\,dx$$

по всевозможным значениям a_i, $0\leqslant i\leqslant m-1$. В точке минимума необходимо

$$\frac{\partial S}{\partial a_i}=0,$$

что эквивалентно соотношению

$$\int_{-1}^{1}x^i l_m(x)\,dx=0,\qquad 0\leqslant i\leqslant m-1,$$

т. е. полином Лежандра $l_m(x)$ ортогонален любому многочлену степени $m-1$. Приведем формулы $l_m(x)$ для $0 \leqslant m \leqslant 4$:

$$l_0(x)=1,$$

$$l_1(x)=x,$$

$$l_2(x)=\frac{1}{3}(3x^2-1),$$

$$l_3(x)=\frac{1}{5}(5x^3-3x),$$

$$l_4(x)=\frac{1}{35}(35x^4-30x^2+3).$$

Общая формула $l_m(x)$ следующая:

$$l_m(x)=x^m-\frac{m(m-1)}{2(2m-1)}x^{m-2}+\frac{m(m-1)(m-2)(m-3)}{2\cdot 4(2m-1)(2m-3)}x^{m-4}-\ldots.$$

Полиномы Лежандра $l_m(x)$ образуют ортогональную систему функций на интервале $[-1, 1]$:

$$\int_{-1}^{1} l_m(x)\,l_n(x)\,dx=0, \qquad m\neq n;$$

их корни просты, вещественны и расположены на интервале $[-1, 1]$.

7.5.2. Квадратурная формула Гаусса. Сформулируем задачу, которую решает квадратурная формула Гаусса.

Требуется для заданного числа узлов, а именно $(m+1)$-го, найти такие узлы ξ_i, $0 \leqslant i \leqslant m$, и соответствующие веса q_i, чтобы квадратурная формула

$$\int_{-1}^{1} f(x)\,dx=\sum_{i=0}^{m} q_i f(\xi_i)+R \tag{7.5.1}$$

была точной ($R=0$) для всех полиномов степени $2m+1$.

Теорема 7.6. *Не существует таких узлов* ξ_i *и весов* q_i, $0 \leqslant i \leqslant m$, *чтобы формула* (7.5.1) *была точной для любого* $P_{2m+2}(x)$.

Доказательство. Предположим противное, т. е. существование ξ_i, q_i для любого $P_{2m+2}(x)$ таких, что $R=0$. Выберем для проверки полином $(2m+2)$-й степени следующего вида:

$$P_{2m+2}(x)=(x-\xi_0)^2(x-\xi_1)^2\ldots(x-\xi_m)^2.$$

Тогда

$$\int_{-1}^{1} P_{2m+2}(x)\,dx>0, \qquad \sum_{i=0}^{m} q_i P_{2m+2}(\xi_i)=0;$$

следовательно, для этого полинома $R\neq 0$. Полученное противоречие доказывает теорему.

Теорема 7.6 устанавливает тот важный факт, что квадратурной формулы, точной на полиномах $P_{2m+2}(x)$, не существует, а следовательно, формула Гаусса должна быть точной на полиномах максимально возможной степени.

Построим формулу Гаусса следующим образом. Будем определять веса q_i в соответствии с (7.1.8):

$$q_i = \int_{-1}^{1} \frac{(x-\xi_0)\dots(x-\xi_{i-1})(x-\xi_{i+1})\dots(x-\xi_m)}{(\xi_i-\xi_0)\dots(\xi_i-\xi_{i-1})(\xi_i-\xi_{i+1})\dots(\xi_i-\xi_m)}\,dx, \tag{7.5.2}$$

где ξ_i — пока произвольные, различные узлы на интервале $[-1, 1]$. Тогда квадратурная формула (7.5.1) будет точной, по крайней мере на полиномах степени m.

Если теперь взять в качестве узлов ξ_i нули полиномов Лежандра $l_{m+1}(x)$

$$l_{m+1}(\xi_i)=0, \quad 0\leqslant i\leqslant m,$$

то по ним можно вычислить веса q_i в соответствии с (7.5.2). Затем по ξ_i, q_i можно определить формулу (7.5.1). Справедлива следующая теорема Гаусса.

Теорема 7.7. *Если в качестве узлов ξ_i, $0\leqslant i\leqslant m$ взяты нули полинома Лежандра $l_{m+1}(x)$, веса вычислены по формуле* (7.5.2), *то квадратурная формула* (7.5.1) *точна для полиномов степени* $2m+1$.

Доказательство. Представим произвольный полином $P_{2m+1}(x)$ в виде

$$P_{2m+1}(x)=\bar{P}_m(x)\,l_{m+1}(x)+\bar{\bar{P}}_m(x), \tag{7.5.3}$$

где $\bar{P}_m(x)$, $\bar{\bar{P}}_m(x)$ — полиномы степени m. Подставим (7.5.3) в левую часть (7.5.1); получим

$$\int_{-1}^{1} P_{2m+1}(x)\,dx=\int_{-1}^{1} \bar{P}_m(x)\,l_{m+1}(x)\,dx+$$

$$+\int_{-1}^{1} \bar{\bar{P}}_m(x)\,dx=\int_{-1}^{1} \bar{\bar{P}}_m(x)\,dx \tag{7.5.4}$$

в силу ортогональности полинома $l_{m+1}(x)$ любому полиному степени m. Подставляя (7.5.3) в правую часть (7.5.1), имеем

$$\sum_{i=0}^{m} q_i P_{2m+1}(\xi_i)=\sum_{i=0}^{m} q_i\left(P_m(\xi_i)\,l_{m+1}(\xi_i)+\right.$$

$$\left.+\bar{\bar{P}}_m(\xi_i)\right)+R=\sum_{i=0}^{m} q_i\bar{\bar{P}}_m(\xi_i)+R \tag{7.5.5}$$

в силу выбора узлов ξ_i. Таким образом, приравнивая (7.5.4) и (7.5.5), находим, что формула (7.5.1) эквивалентна формуле

$$\int_{-1}^{1} \bar{\bar{P}}_m(x)\,dx=\sum_{i=0}^{m} q_i\bar{\bar{P}}_m(\xi_i)+R. \tag{7.5.6}$$

Но если q_i выбраны согласно (7.5.2), то формула (7.5.6) точна для любого полинома $P_m(x)$ степени m; следовательно, $R=0$, что и требовалось доказать.

Можно показать, что корни полиномов Лежандра расположены симметрично относительно нуля, соответствующие веса совпадают, все веса положительны. Приведем значения с тремя знаками для нескольких m узлов ξ_i и весов q_i формул Гаусса:

$m=0$ $(\xi_0=0,\ q_0=2)$;

$m=1$ $(\xi_0=-0{,}577,\ q_0=1{,}000)$, $(\xi_1=0{,}577,\ q_0=1{,}000)$;

$m=2$ $(\xi_0=-0{,}775,\ q_0=0{,}555)$, $(\xi_1=0{,}000,\ q_1=0{,}889)$, $(\xi_2=0{,}775,\ q_0=0{,}555)$;

$m=3$ $(\xi_0=-0{,}861,\ q_0=0{,}348)$, $(\xi_1=-0{,}340,\ q_1=0{,}652)$, $(\xi_2=0{,}340,\ q_2=0{,}652)$, $(\xi_3=0{,}861,\ q_3=0{,}348)$.

Например, квадратурная формула Гаусса по четырем узлам запишется следующим образом:

$$\int_{-1}^{1} f(x)dx=0{,}348\big(f(-0{,}861)+f(0{,}861)\big)+0{,}652\big(f(-0{,}340)+f(0{,}340)\big)+R.$$

7.5.3 Оценка погрешности формулы Гаусса. Пусть используется квадратурная формула Гаусса с узлами ξ_i, $0\leqslant i\leqslant m$. Будем предполагать, что подынтегральная функция $f(x)\in C^{2(m+1)}$ на интервале $[-1, 1]$. Тогда можно показать, что погрешность $R(m, f)$ (7.5.1) имеет вид

$$R(m, f)=\frac{2^{2(m+1)+1}\big[(2m+1)!\big]^4}{\big[2(m+1)!\big]^3\big[2(m+1)+1\big]}f^{2(m+1)}(\xi), \qquad (7.5.7)$$

где $-1\leqslant\xi\leqslant 1$. Выражение (7.5.7), так же как аналогичные соотношения для простейших и составных квадратурных формул, имеет в основном теоретическое значение.

На практике широко применяется сравнение результатов вычислений на нескольких семействах узлов. Произвольный интервал интегрирования $[a, b]$ разбивается шагом h на n частичных интервалов $[x_i, x_{i+1}]$, $0\leqslant i\leqslant n-1$. Каждый из частичных интервалов заменой независимого переменного сводится к интервалу $[-1, 1]$, затем производится вычисление по квадратурной формуле (7.5.1) с $m+1$ узлами. Результаты суммируются. Это алгоритм составной квадратурной формулы Гаусса.

Варьируя параметрами m, n и сравнивая результаты вычислений, можно оценить погрешность составной квадратурной формулы Гаусса.

7.5.4. Применение программы A4A1. В программе A4A1 реализован алгоритм составной квадратурной формулы Гаусса со значением $m=9$ и перебором по n от 1 до 10. Производится контроль заданной абсолютной и относительной точности вычисления интеграла. Пусть необходимо вычислить интеграл

$$\int_0^3 \sin\left(e^{-x^2}\right)dx$$

с абсолютной точностью 10^{-5} и относительной 10^{-4}. Программа может иметь следующий вид:

```
      REAL A,B,E,D,Y,E1
      EXTERNAL F
      DATA A,B,E,D/0.,3.,1.E—5,1.E—4/
C     ОБРАЩЕНИЕ К ПРОГРАММЕ A4A1
      CALL A4A1(F,A,B,E,D,Y,E1)
C     ВЫВОД НА ТЕРМИНАЛ
      WRITE (5,1) Y,E1
1     FORMAT (2X,'Y=',E13.6,'E1=',E13.6)
      END
C     ВНЕШНЯЯ ФУНКЦИЯ-ПОДПРОГРАММА
      FUNCTION F(X)
      F=SIN(EXP(—(X*X)))
      RETURN
      END
```

● 7.6. Интегрирование функций двух переменных

При численном интегрировании функций двух переменных $f(x, y)$ по некоторым областям D часто применяются квадратурные формулы, которые получаются комбинацией одномерных квадратурных формул.

7.6.1 Формула Гаусса для квадрата. Рассмотрим интегрирование $f(x, y)$ по областям D:

$$D=\{-1\leqslant x\leqslant 1, \quad -1\leqslant y\leqslant 1\}. \tag{7.6.1}$$

Произвольный прямоугольник $\{a\leqslant x\leqslant b, \quad c\leqslant y\leqslant d\}$ заменой x, y следует привести к стандартному квадрату (7.6.1). Возьмем по переменной x узлы ξ_i и соответствующие веса q_i, $0\leqslant i\leqslant m$, квадратурной формулы Гаусса одной переменной, по y — узлы η_i и соответствующие веса r_j, $0\leqslant j\leqslant n$. На рис. 7.13 показан вариант $m=3$, $n=2$. Составим квадратурную формулу

$$Q_{m,n}=\sum_{i,j=0}^{m,n} q_i r_j f(\xi_i, \eta_j) \tag{7.6.2}$$

и представим интеграл от $f(x, y)$ по области D в виде

$$\int_{-1}^{1}\int_{-1}^{1} f(x, y)\,dxdy=Q_{m,n}+R_{m,n}(f), \tag{7.6.3}$$

где погрешность R зависит от m, n и свойств гладкости $f(x, y)$ по переменным x, y.

Заметим, что квадратурная формула (7.6.2) является точной на полиномах от двух переменных:

$$P_{2m+1,\,2n+1}(x, y)=\sum_{i,j=0}^{2m+1,\,2n+1} a_{ij}x^i y^j. \tag{7.6.4}$$

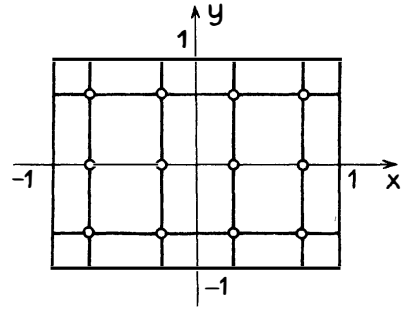

Рис. 7.13

Отмеченный факт является следствием теоремы 7.7. Для примера, изображенного на рис. 7.13, это полиномы

$$P_{7,5}(x, y)=a_{0,0}+a_{1,0}x+a_{2,0}x^2+a_{3,0}x^3+\ldots+a_{7,0}x^7+$$
$$+a_{0,1}y+a_{1,1}xy+a_{2,1}x^2y+a_{3,1}x^3y+\ldots+a_{7,1}x^7y+$$
$$+\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots+$$
$$+a_{0,5}y^5+a_{1,5}xy^5+a_{2,5}x^2y^5+a_{3,5}x^3y^5+\ldots+a_{7,5}x^7y^5$$

Теорема 7.8. *Пусть подынтегральная функция* $f(x, y)$ *аппроксимируется полиномом* (7.6.4) *и выполняется неравенство*

$$\max_{-1\leqslant x,\, y\leqslant +1} |f(x, y)-P_{2m+1,2n+1}(x, y)|\leqslant\varepsilon.$$

Тогда погрешность формулы (7.6.2) *на этой функции имеет оценку*

$$|R_{m,n}(f)|\leqslant 8\varepsilon. \tag{7.6.5}$$

Доказательство. Действительно, представим из (7.6.3) $R_{m,n}(f)$ в виде

$$R_{m,n}(f)=\int_{-1}^{1}\int_{-1}^{1}(f-P_{2m+1,2n+1}+P_{2m+1,2n+1})\,dxdy-$$
$$-\sum_{i,j=0}^{m,n}q_ir_j(f(\xi,\eta_j)-P_{2m+1,2n+1}(\xi_i,\eta_j)+P_{2m+1,2n+1}(\xi_i,\eta_j)).$$

Отсюда

$$|R_{m,n}(f)|\leqslant\int_{-1}^{1}\int_{-1}^{1}|f-P_{2m+1,2n+1}|\,dxdy+$$
$$+\sum_{i,j=0}^{m,n}q_ir_j|f-P_{2m+1,2n+1}|\leqslant 4\varepsilon+\varepsilon\sum_{i,j=0}^{m,n}q_ir_j.$$

Формула (7.6.2) точна на константах, поэтому

$$\int_{-1}^{1}\int_{-1}^{1}dxdy=\sum_{i,j=0}^{m,n}q_ir_j=4;$$

следовательно, учитывая последнее неравенство, получаем (7.6.5), что и требовалось доказать.

7.6.2. Область со сложной границей. Рассмотрим задачу численного интегрирования $f(x, y)$ по области D:

$$D=\{\varphi_1(y)\leqslant x\leqslant\varphi_2(y),\quad c\leqslant y\leqslant d\},$$

где часть границы определяется непрерывными функциями $\varphi_1(y)$, $\varphi_2(y)$ (рис. 7.14).

Требуется вычислить интеграл

$$I=\iint_D f(x, y)\,dxdy=\int_c^d dy\int_{\varphi_1(y)}^{\varphi_2(y)}f(x, y)\,dx. \tag{7.6.6}$$

Сведение двойного интеграла к повторному (7.6.6) позволяет дважды применить одномерные квадратурные формулы (например, формулы Гаусса). Исходный интеграл запишем в виде

$$v(y)=\int_{\varphi_1(y)}^{\varphi_2(y)} f(x, y)\,dx, \quad (7.6.7)$$

$$Y=\int_c^d v(y)\,dy. \quad (7.6.8)$$

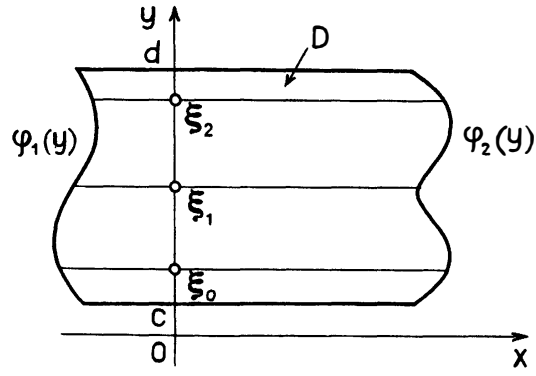

Рис. 7.14

Интеграл (7.6.8) вычислим с помощью одномерной формулы Гаусса:

$$Y=\int_c^d v(y)\,dy=\sum_{j=0}^{n} r_j v(\eta_j)+R_1; \quad (7.6.9)$$

веса r_j и узлы η_j определяются, как указывалось выше, приведением к стандартному интервалу $[-1, 1]$. Затем каждое значение $v(\eta_j)$ вычислим, используя (7.6.7), также по одномерной формуле Гаусса:

$$v(\eta_j)=\int_{\varphi_1(\eta_j)}^{\varphi_2(\eta_j)} f(x, \eta_j)\,dx=\sum_{i=0}^{m} q_{i,j} f(\xi_{i,j}, \eta_j)+R_2, \quad (7.6.10)$$

где веса $q_{i,j}$ и узлы $\xi_{i,j}$ зависят от выбора узлов η_j. Объединяя формулы (7.6.10), получим формулу Гаусса для рассматриваемой области D:

$$I=Q+R,$$

$$Q=\sum_{j=0}^{n} r_j \sum_{i=0}^{m} q_{i,j} f(\xi_{i,j}, \eta_j), \quad (7.6.11)$$

где R — общая погрешность, зависящая от числа узлов и свойств гладкости функции $f(x, y)$.

7.6.3. Применение программы A4A2. В программе реализован алгоритм вычисления интеграла (7.6.6) по формуле (7.6.11) с выбором числа узлов по заданной абсолютной погрешности вычисления I.

Пусть необходимо вычислить

$$I=\int_0^1 dy \int_{\cos y-2}^{\sin y+1} \exp(x^2+y^2)\,dx$$

с абсолютной точностью 10^{-4}. Программа может иметь следующий вид:

```
REAL A,B,E,Z
INTEGER N,I
EXTERNAL F1,F2,F
DATA A,B/0.,1./,E/1.E—4/,I/0/
```

```
C       ОБРАЩЕНИЕ К ПРОГРАММЕ A4A2
        CALL A4A2(A,B,F1,F2,F,E,Z,N,I)
C       ВЫВОД НА ТЕРМИНАЛ
        WRITE (5,1) Z,N,I
1       FORMAT (2X,'Z=',E13.6,'N=',15,'I=',12)
        END
C       ВНЕШНИЕ ФУНКЦИИ-ПОДПРОГРАММЫ
        FUNCTION F1(Y)
        F=COS(Y)-2.
        RETURN
        END
        FUNCTION F2(Y)
        F2=SIN(Y)+1.
        RETURN
        END
        FUNCTION F(X,Y)
        F=EXP(X*X+Y*Y)
        RETURN
        END
```

Глава 8

ЛИНЕЙНАЯ АЛГЕБРА. ЛИНЕЙНАЯ ОПТИМИЗАЦИЯ

● 8.1. Оценки погрешности решения задач линейной алгебры

8.1.1. Введение. Численные методы линейной алгебры и линейной оптимизации играют особую роль в численном анализе. Это обусловлено по крайней мере двумя причинами.

Во-первых, многие линейные задачи математического анализа, дифференциальных и интегральных уравнений после дискретизации сводятся к решению задач линейной алгебры. Таким образом, численные методы линейной алгебры оказываются инструментом численного решения обширного круга математических, а следовательно, научно-технических задач.

Во-вторых, можно сформулировать следующее нестрогое утверждение: большинство нелинейных задач «в малом» линейны, т. е. нелинейные модели в малой окрестности некоторого решения могут быть описаны линейными. В основе конечномерных линейных моделей лежит линейная алгебра; следовательно, первым шагом решения нелинейных задач является исследование линеаризованных моделей, их дискретизация и численные методы линейной алгебры.

Однако, как нельзя недооценивать роль линейной алгебры и ее численных методов, так и не следует переоценивать ее среди инструментов решения научно-технических задач. Реальные математические модели все-таки нелинейны, обычно необходимо изучить модель «в целом», а не «в малом». Следовательно, полный арсенал методов вычислений должен содержать нелинейный численный анализ.

В гл. 5 отмечалось, что потребность в численных методах линейной алгебры связана в основном с размерностью задачи. Для малых размерностей n конечномерного пространства ($n \leqslant 4$) задачи линейной алгебры решаются аналитически на уровне формул.

Естественно, что численные методы применимы как для размерностей $n \geqslant 5$, так и для малых размерностей $n = 2, 3$. Поэтому все основные проблемы численного решения линейных задач можно и нужно представлять наглядно, геометрически, на плоскости или трехмерном пространстве. Численный метод, представленный геометрически в малой размерности, имеет все основные черты, присущие ему и в большой размерности, отличаясь лишь меньшим объемом арифметических операций. Есть, конечно, проблемы, связанные именно с размерностью n (ограничение памяти ЭВМ, ошибки длинных вычислений); их следует рассматривать отдельно.

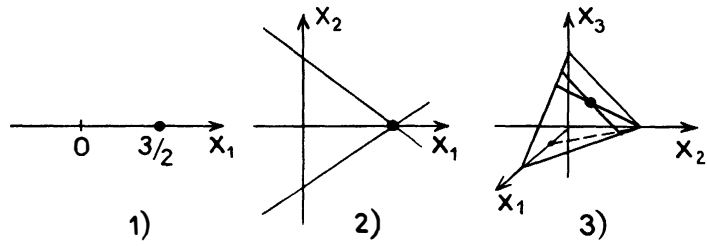

Рис. 8.1

8.1.2. Оценка погрешности решения систем линейных уравнений. Система линейных уравнений с матрицей вещественных коэффициентов $A=(a_{i,j})$, $1\leqslant i,j\leqslant n$, и вектором свободных членов $b=(b_1, ..., b_n)$ относительно неизвестного вектора $x=(x_1, ..., x_n)$ записывается следующим образом:

$$\begin{aligned} &a_{1,1}x_1+a_{1,2}x_2+...+a_{1,n}x_n=b_1,\\ &a_{2,1}x_1+a_{2,2}x_2+...+a_{2,n}x_n=b_2,\\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ &a_{n,1}x_1+a_{n,2}x_2+...+a_{n,n}x_n=b_n, \end{aligned} \tag{8.1.1}$$

или в матричной форме

$$Ax=b. \tag{8.1.2}$$

Каждое уравнение (8.1.1) описывает прямую ($n=2$), плоскость ($n=3$), гиперплоскость ($n\geqslant 4$) в вещественном пространстве E^n, поэтому решить (8.1.1) — значит найти точку $x\in E^n$ их пересечения.

Например (рис. 8.1):

1) $2x_1=3$;
2) $x_1+x_2=1$;
   $x_1-x_2=1$;
3) $x_1+x_2+x_3=1$,
   $2x_1+x_2=1$
   $x_2+x_3=0{,}8$.

На рис. 8.1, 3) показана первая плоскость и на ней следы двух других плоскостей.

Прежде чем оценивать погрешность решения (8.1.1), оценим погрешность операции умножения матрицы на вектор:

$$y=Ax. \tag{8.1.3}$$

Будем предполагать, что матрица A задана точно, а вектор x — с погрешностью Δx такой, что

$$\|\Delta x\|\leqslant r. \tag{8.1.4}$$

Множество точек $x=x_0+\Delta x$, удовлетворяющих (8.1.4), называется шаром в E^n радиуса r с центром в x_0. На рис. 8.2 приведены примеры шаров для некоторых норм в E^2.

Чтобы найти абсолютную погрешность Δy вектора y, запишем (8.1.3) в виде

$$y+\Delta y=A(x+\Delta x).$$

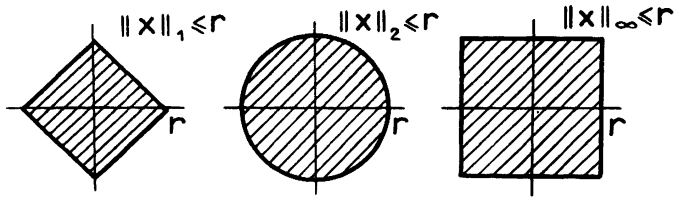

Рис. 8.2

Отсюда получаем связь вектора погрешности Δy к точному значению y с вектором погрешности Δx вектора x:

$$\Delta y = A\Delta x.$$

Из определения нормы матрицы находим

$$\|\Delta y\| = \|A\Delta x\| \leqslant \|A\| \|\Delta x\| \leqslant \|A\| r. \tag{8.1.5}$$

Напомним, что норму матрицы A можно задать формулой (эквивалентной определению п. 5.3.10)

$$\|A\| = \max_{\|x\|=1} \|Ax\|.$$

Геометрическая интерпретация нормы матрицы — максимальная норма вектора, получаемого преобразованием A единичной сферы $\|x\|=1$ (рис. 8.3). Неравенство (8.1.5) показывает, что радиус шара погрешности вектора y изменяется в $\|A\|$ раз.

Теорема 8.1. *Пусть система линейных уравнений* (8.1.2) *для любого вектора* b *имеет единственное решение. Тогда имеет место неравенство*

$$\|\Delta x\| \leqslant \|A^{-1}\| \|\Delta b\|, \tag{8.1.6}$$

где Δb, Δx — векторы абсолютной погрешности b, x соответственно.

Доказательство. В силу предположения однозначной разрешимости (8.1.2) для любого b существует матрица A^{-1}, обратная матрице A. Используя A^{-1}, запишем решение (8.1.2):

$$x = A^{-1}b. \tag{8.1.7}$$

Применяя к (8.1.7) неравенство (8.1.5), получаем (8.1.6), что и требовалось доказать.

В качестве примера рассмотрим систему уравнений (8.1.2) с двумя матрицами второго порядка

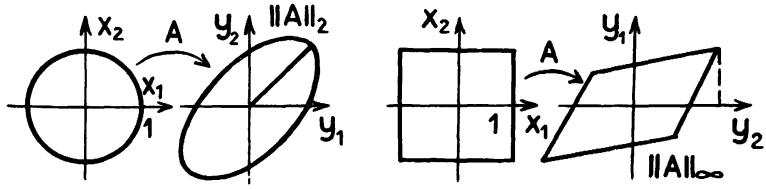

Рис. 8.3

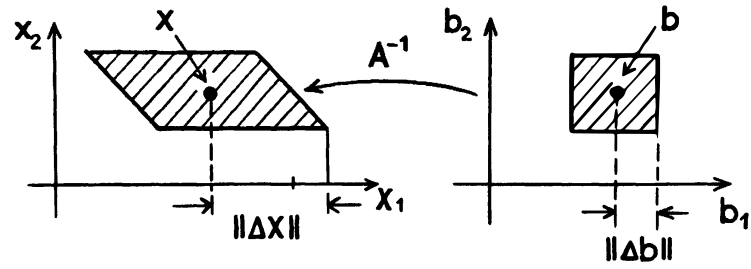

Рис. 8.4

$$A_1=\begin{pmatrix}10 & 3\\ 0 & 10\end{pmatrix},\quad A_2=\begin{pmatrix}0{,}1 & 3\\ 0 & 0{,}1\end{pmatrix},$$

для которых легко определить обратные матрицы

$$A_1^{-1}=\begin{pmatrix}0{,}1 & -0{,}03\\ 0 & 0{,}1\end{pmatrix},\quad A_2^{-1}=\begin{pmatrix}10 & -300\\ 0 & 10\end{pmatrix}.$$

В качестве векторной и соответственно матричной нормы возьмем

$$\|x\|_\infty=\max_{1\leqslant i\leqslant n}|x_i|,\qquad \|A\|_\infty=\max_{1\leqslant i\leqslant n}\sum_{j=1}^{n}|a_{i,j}|$$

Погрешность решения первой системы имеет оценку

$$\|\Delta x\|\leqslant 0{,}13\,\|\Delta b\|,$$

второй — оценку

$$\|\Delta x\|\leqslant 310\,\|\Delta b\|,$$

причем знак равенства достигается на векторе Δb с координатами разных знаков (рис. 8.4).

Для характеристики чувствительности системы линейных уравнений (8.1.2) к погрешностям правых частей вводятся числа обусловленности.

Число обусловленности системы μ определяется как максимальное отношение относительной погрешности решения $\delta_x=\dfrac{\|\Delta x\|}{\|x\|}$ к относительной погрешности правой части $\delta_b=\dfrac{\|\Delta b\|}{\|b\|}$:

$$\mu=\max\frac{\delta_x}{\delta_b}.$$

Число обусловленности системы μ можно выразить через $\|A^{-1}\|$. Действительно, имеем

$$\mu=\max\frac{\|\Delta x\|\,\|b\|}{\|x\|\,\|\Delta b\|}=\frac{\|b\|}{\|x\|}\max\frac{\|\Delta x\|}{\|\Delta b\|}=\|A^{-1}\|\,\frac{\|b\|}{\|x\|}.$$

Заметим, что $\mu=\mu(b)$ зависит от вектора в правой части и решения x, соответствующего b. Второе число обусловленности характеризует только матрицу системы A.

Число обусловленности ν *матрицы* A есть максимальное по всем векторам b значение $\mu(b)$:

$$\nu = \max_b \mu(b).$$

Число обусловленности ν можно записать в следующей форме (с учетом $\|b\| \leqslant \|A\| \|x\|$):

$$\nu = \max_b \|A^{-1}\| \frac{\|b\|}{\|x\|} = \|A^{-1}\| \|A\|.$$

Из цепочки соотношений

$$1 = \|E\| = \|A^{-1}A\| \leqslant \|A^{-1}\| \|A\| = \nu,$$

где E — единичная матрица, следует, что $\nu \geqslant 1$.

Для приведенных выше примеров $\nu_1 = 1{,}69$, $\nu_2 = 961$.

Рассмотрим задачу решения системы уравнений с матрицей A_2 и различными векторами $b = (b_1, b_2)$, компоненты которых задаются с пятью верными значащими цифрами. Тогда значение $\nu_2 = 961 \simeq 10^3$ указывает, что решение x этих систем можно вычислить лишь с двумя верными значащими цифрами.

Для конкретной правой части, например $b = (1, 0)$, имеем $x = (10, 0)$, число $\mu_2 = 31$. Оно указывает, что решение x этой системы можно вычислить не менее чем с тремя верными значащими цифрами.

Оценим теперь погрешность решения исходной системы уравнений (8.1.2) в том случае когда матрица системы A задается с погрешностью ΔA, а вектор b — точно.

Теорема 8.2. *Пусть система линейных уравнений* (8.1.2) *для любого вектора* b *имеет единственное решение. Пусть*

$$\|A^{-1}\Delta A\| \leqslant \alpha < 1. \tag{8.1.8}$$

Тогда (8.1.2) *однозначно разрешима с матрицей* $(A + \Delta A)$ *и имеет место неравенство*

$$\frac{\|\Delta x\|}{\|x\|} \leqslant \frac{\nu}{1-\alpha} \frac{\|\Delta A\|}{\|A\|}. \tag{8.1.9}$$

Доказательство. Представим уравнение (8.1.2) в виде

$$(A + \Delta A)(x + \Delta x) = b.$$

Из этого равенства находим

$$\begin{aligned} A(E + A^{-1}\Delta A)(x + \Delta x) &= b, \\ (E + A^{-1}\Delta A)(x + \Delta x) &= A^{-1}b. \end{aligned} \tag{8.1.10}$$

Из условия (8.1.8) следует существование обратной матрицы $(E + A^{-1}\Delta A)^{-1}$ и ее представление (см. гл. 5)

$$(E + A^{-1}\Delta A)^{-1} = \sum_{i=0}^{\infty} (A^{-1}\Delta A). \tag{8.1.11}$$

Подставляя (8.1.11) в (8.1.10), находим

$$x+\Delta x=A^{-1}b+\sum_{i=1}^{\infty}\left(A^{-1}\Delta A\right)^{i}A^{-1}b,$$

отсюда

$$\Delta x=A^{-1}\Delta A\sum_{i=0}^{\infty}\left(A^{-1}\Delta A\right)^{i}A^{-1}b.$$

Переходя к оценкам норм, получаем

$$\|\Delta x\|\leqslant\|A^{-1}\|\|\Delta A\|\left(\sum_{i=0}^{\infty}\|A^{-1}\Delta A\|^{i}\right)\|x\|\leqslant$$

$$\leqslant\frac{\|x\|\|A^{-1}\|\|\Delta A\|}{1-\alpha}=\frac{\|x\|\|A^{-1}\|\|A\|\|\Delta A\|}{(1-\alpha)\|A\|},$$

откуда следует (8.1.9), что и требовалось доказать.

Можно получить оценку относительной погрешности решения (8.1.2) в общем случае, когда матрица A и вектор b задаются с погрешностью ΔA и Δb соответственно.

Формулировка теоремы 8.2 остается прежней, а оценка (8.1.9) заменяется следующей:

$$\frac{\|\Delta x\|}{\|x\|}\leqslant\frac{\nu}{1-\alpha}\left(\frac{\|\Delta A\|}{\|A\|}+\frac{\|\Delta b\|}{\|b\|}\right). \tag{8.1.12}$$

Неравенство (8.1.12) показывает, что влияние погрешности матрицы A и вектора b на погрешность решения связано с двумя числами α, ν.

Число α определяет допустимую погрешность матрицы A, которая еще сохраняет однозначную резрешимость систем (8.1.2). Для рассмотренных выше примеров $0{,}13\|\Delta A\|=\alpha_1$, $310\|\Delta A\|=\alpha_2$. Отсюда получаем ограничение на допустимую погрешность матрицы системы: для первой системы $\|\Delta A_1\|<13$, для второй $\|\Delta A_2\|<(310)^{-1}$, или в относительных погрешностях

$$\frac{\|\Delta A_1\|}{\|A_1\|}<1,\qquad\frac{\|\Delta A_2\|}{\|A_2\|}<\frac{1}{310\cdot 3{,}1}=\frac{1}{961}.$$

Естественно задать вопрос: что делать, если величина $\nu(1-\alpha)^{-1}$ столь велика, что погрешность $\dfrac{\|\Delta x\|}{\|x\|}$ не удовлетворяет заданной точности вычислений? В этом случае необходимо обратиться к постановке той задачи, которая привела к решению такой системы линейных уравнений (малые погрешности исходных данных могут приводить к неприемлемо большим погрешностям в решении), учесть дополнительную информацию в задаче так, чтобы величина в правой части (8.1.12) стала приемлемой.

В гл. 6 мы, в принципе, так и поступали, когда предлагали заменить в алгоритме среднеквадратичного приближения семейство полиномов $\{x^i\}$ полиномами Чебышева $\{T_i(x)\}$.

Если матрица A и вектор b определяются из экспериментальных данных, то оценка (8.1.12), возможно, станет приемлемой, если провести измерения с более высокой точностью.

Заметим, что в приведенных выше оценках погрешности не учитывались ни погрешности метода решения системы, ни погрешности вычислительного процесса по выбранному методу.

Фактически это оценки погрешности математической модели, описываемой системой (8.1.2).

8.1.3. Оценка погрешности определения собственных значений и собственных векторов. Пусть задача определения собственных значений и собственных векторов вещественной матрицы A решается в том случае, когда матрица A задается с погрешностью ΔA. Оценим погрешность собственных значений и собственных векторов через погрешность ΔA.

Пусть матрица A имеет различные собственные значения λ_1, λ_2, ..., λ_n и соответствующие им собственные векторы e_1, e_2, ..., e_n. Известно, что $\{e_i\}$, $1 \leqslant i \leqslant n$, образуют базис в E^n, а матрица T, столбцами которой являются компоненты векторов e_i

$$T = \begin{pmatrix} e_{1,1} & \dots & e_{n,1} \\ & \dots & \\ e_{1,n} & \dots & e_{n,n} \end{pmatrix}, \tag{8.1.13}$$

приводит исходную матрицу A к диагональному виду, а именно

$$\operatorname{diag}(\lambda_i) = \begin{pmatrix} \lambda_1 & \cdot & 0 \\ \cdot & \cdot & \cdot \\ 0 & \cdot & \lambda_n \end{pmatrix} = T^{-1}AT.$$

Собственные значения и собственные векторы, соответствующие матрице $A + \Delta A$, обозначим $\lambda_i + \Delta\lambda_i$, $e_i + \Delta e_i$, $1 \leqslant i \leqslant n$. Если применить к матрице $A + \Delta A$ преобразование T, то получим

$$T^{-1}(A + \Delta A)T = \operatorname{diag}(\lambda_i) + T^{-1}\Delta AT. \tag{8.1.14}$$

Известно, что матрица $T^{-1}(A + \Delta A)T$ имеет те же собственные значения, что и $(A + \Delta A)$. Воспользовавшись критерием локализации собственных значений (5.3.15), получим из (8.1.14) неравенство

$$|(\lambda_i + \Delta\lambda_i) - \operatorname{diag}(\lambda_i) - \operatorname{diag}(T^{-1}\Delta AT)| \leqslant \sum_{j=1}^{n} |(T^{-1}\Delta AT)_{i,j}|, \quad 1 \leqslant i \leqslant n,$$

или

$$|\Delta\lambda_i| \leqslant \| T^{-1}\Delta AT \| \leqslant \| T^{-1} \| \, \| T \| \, \| \Delta A \|,$$

Здесь $\| \ \| = \| \ \|_\infty$. Обозначим число обусловленности ν матрицы T

$$\nu(T) = \| T^{-1} \| \, \| T \|.$$

Окончательно имеем оценку

$$|\Delta\lambda_i| \leqslant \nu(T) \| \Delta A \|, \quad 1 \leqslant i \leqslant n, \tag{8.1.15}$$

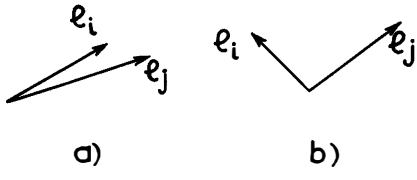

Рис. 8.5

которая показывает, что абсолютная погрешность определения собственных чисел зависит от числа обусловленности матрицы T преобразования A к диагональному виду.

Заметим, что неравенство (8.1.15) получено только при предположении существования различных собственных значений. Точных значений λ_i, векторов e_i знать не обязательно.

Для произвольных матриц величина $\nu(T)$ может принимать большие значения, если имеются собственные векторы A, близкие к линейно зависимым (рис. 8.5, *a*). Для симметричных матриц A собственные векторы e_i образуют ортонормированный базис (рис. 8.5, *b*), и если $(A+\Delta A)$ также симметрична, то можно надеяться на хорошую устойчивость задачи определения собственных значений симметричных матриц относительно погрешностей, которые не нарушают симметричности матрицы.

Рассмотрим задачу определения поправок к собственным значениям и векторам симметричной матрицы A, связанных с погрешностью ΔA, как задачу возмущений (см. гл. 5). Для этого представим

$$\Delta A = \varepsilon B,$$

где ε — параметр возмущения, характеризующий погрешность, B — симметричная матрица. В этом случае собственные значения и векторы матрицы $(A+\varepsilon B)$ есть функции от ε вида

$$\lambda_i(\varepsilon) = \lambda_i + \varepsilon\lambda^{(1)} + \varepsilon^2\lambda_i^{(2)} + \ldots; \tag{8.1.16}$$

$$e_i(\varepsilon) = e_i + \varepsilon e_i^{(1)} + \varepsilon^2 e_i^{(2)} + \ldots, \tag{8.1.17}$$

где ряды сходятся для достаточно малых ε (при наличии различных собственных значений λ_i матрицы A). Доказательство сходимости рядов выходит за рамки настоящей книги.

Теорема 8.3. *Пусть вещественная симметричная матрица A имеет собственные значения λ_i, $1 \leqslant i \leqslant n$, и векторы e_i. Тогда, если B — симметричная матрица, собственные значения $\lambda_i(\varepsilon)$, $1 \leqslant i \leqslant n$, и векторы $e_i(\varepsilon)$ матрицы $(A+\varepsilon B)$ имеют при $\varepsilon \to 0$ представление*

$$\lambda_i(\varepsilon) = \lambda_i + \varepsilon(Be_i, e_i) + O(\varepsilon^2); \tag{8.1.18}$$

$$e_i(\varepsilon) = e_i + \varepsilon \sum_{\substack{j=1 \\ i \neq j}}^{n} \frac{(Be_i, e_j)}{(\lambda_i - \lambda_j)} + O(\varepsilon^2) \tag{8.1.19}$$

Доказательство. Найдем главную часть поправки в (8.1.16), (8.1.17) к невозмущенным собственным значениям λ_i и векторам e_i матрицы A, т. е. $\lambda_i^{(1)}$, $e_i^{(1)}$. Имеем

$$(A+\varepsilon B)e_i(\varepsilon) = \lambda_i(\varepsilon)e_i(\varepsilon), \quad 1 \leqslant i \leqslant n,$$

или с учетом (8.1.16), (8.1.17)

$$(A+\varepsilon B)\left(e_i+\varepsilon e_i^{(1)}+O(\varepsilon^2)\right)=\left(\lambda_i+\varepsilon\lambda_i^{(1)}+O(\varepsilon^2)\right)\times$$
$$\times\left(e_i+\varepsilon e_i^{(1)}+O(\varepsilon^2)\right),\quad 1\leqslant i\leqslant n.$$

Следуя общей идее методов возмущений, сравним коэффициенты при одинаковых степенях ε.

При ε°

$$A\varepsilon_i=\lambda_i e_i,\quad 1\leqslant i\leqslant n.$$

При ε^1

$$Ae_i^{(1)}+Be_i=\lambda_i e_i^{(1)}+\lambda_i^{(1)}e_i,\quad 1\leqslant i\leqslant n. \tag{8.1.20}$$

Умножим обе части последнего равенства скалярно на e_i, а также учтем, что векторы e_i образуют ортонормированный базис, а матрица A — вещественная, симметричная. Находим

$$(Ae_i^{(1)},\ e_i)+(Be_i,\ e_i)=\lambda_i(e_i^{(1)},\ e_i)+\lambda_i^{(1)}(e_i,\ e_i),$$
$$(Ae_i^{(1)},\ e_i)=(e_i^{(1)},\ Ae_i)=\lambda_i(e_i^{(1)},\ e_i),$$

откуда

$$(Be_i,\ e_i)=\lambda_i^{(1)},\quad 1\leqslant i\leqslant n.$$

Следовательно, (8.1.18) выполняется.

Умножим обе части равенства (8.1.20) на e_j, $j\neq i$, получим

$$(Ae_i^{(1)},\ e_j)+(Be_i,\ e_j)=\lambda_i(e_i^{(1)},\ e_j),$$
$$(Ae_i^{(1)},\ e_j)=(e_i^{(1)},\ Ae_j)=\lambda_j(e_i^{(1)},\ e_j),$$

откуда

$$(e_i^{(1)},\ e_j)=\frac{1}{\lambda_i-\lambda_j}(Be_i,\ e_j),\quad i\neq j. \tag{8.1.21}$$

Формулы (8.1.21) определяют коэффициенты разложения векторов $e_i^{(1)}$ по базису e_j, $1\leqslant j\leqslant n$, для всех векторов, кроме случая $j=i$. Для определения коэффициента $(e_i^{(1)},\ e_i)$ воспользуемся условием нормировки векторов $e_i(\varepsilon)$:

$$\left(e_i+\varepsilon e_i^{(1)}+O(\varepsilon^2),\ e_i+\varepsilon e_i^{(1)}+O(\varepsilon^2)\right)=1.$$

Сравнивая члены при степенях ε в этом равенстве, находим

$$(e_i^{(1)},\ e_i)+(e_i,\ e_i^{(1)})=0,$$

или

$$(e_i^{(1)},\ e_i)=0. \tag{8.1.22}$$

Объединяя (8.1.21) и (8.1.22) в одну формулу, получаем

$$e_i^{(1)}=\sum_{j=1}^{n}(e_i^{(1)},\ e_j)e_j=\sum_{\substack{j=1\\ j\neq 1}}^{n}\frac{(Be_i,\ e_j)}{\lambda_i-\lambda_j}e_j,\quad 1\leqslant i\leqslant n$$

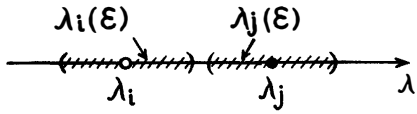

Рис. 8.6

Следовательно, справедливо соотношение (8.1.19), что и требовалось доказать.

Практически формулы (8.1.18), (8.1.19) используют при конечных значениях ε, а не при $\varepsilon \to 0$. Допустимые значения ε определяются такими значениями, при которых возмущенные собственные значения $\lambda_i(\varepsilon)$ не пересекаются (рис. 8.6). При этом поправки к собственным векторам должны быть такими, чтобы имело место неравенство

$$\varepsilon(Be_i, e_j) \ll |\lambda_i - \lambda_j|,$$

что следует из (8.1.19) (знак $\ll$ означает, что левая часть значительно меньше правой).

8.1.4. Вырожденные матрицы и кратные собственные значения. Выше, рассматривая оценки погрешности решения систем линейных уравнений, мы предполагали, что система линейных уравнений

$$Ax = b$$

имеет единственное решение для любой правой части. Это предположение эквивалентно тому, что матрица A невырождена, т. е.

$$\det A \neq 0.$$

Оценка погрешности определения собственных значений проводилось при условии, что собственные значения λ_i, $1 \leqslant i \leqslant n$, различны, т. е. нет кратных λ_i, а именно

$$\lambda_i \neq \lambda_j, \quad i \neq j.$$

Рассмотрим вопрос о том, насколько обоснованы эти предположения. Не отбрасываются ли при этом важные прикладные задачи с вырожденными матрицами или кратными собственными значениями? Заметим, что любая вещественная матрица A размера $n \times n$ может рассматриваться как элемент вещественного пространства E^{n^2}. Каждая точка E^{n^2} задает некоторую матрицу A.

Вырожденные матрицы A^* в E^{n^2} выделяются условием

$$\det A^* = 0. \tag{8.1.23}$$

Матрицы A, удовлетворяющие (8.1.23), представляют собой множество в E^{n^2} размерности, меньшей n^2.

Например, для матриц второго порядка

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix}$$

вырожденные матрицы A^* в пространстве E^4, образованном векторами $(a_{1,1}, a_{1,2}, a_{2,1}, a_{2,2})$, выделяются условием

$$a^*_{1,1} a^*_{2,2} - a^*_{1,2} a^*_{2,1} = 0.$$

Таким образом, произвольное, сколь угодно малое возмущение матрицы A^* в E^{n^2} должно приводить к нарушению условия (8.1.23) (рис. 8.7).

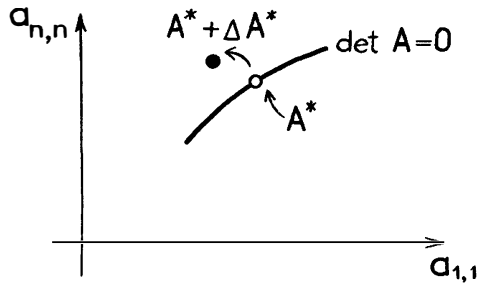

Рис. 8.7

Следовательно, можно утверждать, что вырожденные матрицы в семействе матриц $n \times n$ представляют собой исключение, свойство вырожденности является неустойчивым. Поэтому большинство технических задач приводится к уравнениям с невырожденными матрицами.

Если же математическая модель задачи оказалась связанной с вырожденной системой линейных уравнений или близкой к ней, то следует изменить модель. Если оказывается, что математическая модель в рамках принятых предположений все-таки определяется матрицей, близкой к вырожденной, то, видимо, техническая модель находится к неустойчивой ситуации со всеми вытекающими отсюда последствиями. Это является сигналом к изменению технического решения.

Кратные собственные значения λ имеют матрицы A, у которых характеристическое уравнение

$$P_n(\lambda)=\det(A-\lambda E)=0$$

с кратными корнями. Но если полином $P_n(\lambda)$ имеет λ_* — кратный корень, то с необходимостью должно выполняться условие

$$\frac{dP_n}{d\lambda}(\lambda_*)=0. \qquad (8.1.24)$$

Условие (8.1.24), так же как (8.1.23), из пространства E^{n^2} выделяет множество матриц A^* меньшей размерности.

Например, для матриц второго порядка

$$P_2(\lambda)=(a_{1,1}-\lambda)(a_{2,2}-\lambda)-a_{2,1}a_{1,2}$$

корни

$$\lambda_{1/2}=\frac{a_{1,1}+a_{2,2}}{2}\pm\sqrt{\left(\frac{a_{1,1}+a_{2,2}}{4}\right)^2-\det A}$$

кратны при условии (8.1.24), которое принимает вид

$$(a_{1,1}+a_{2,2})^2-4\det A=0.$$

Поэтому относительно матриц A с кратными собственными значениями справедливы те же замечания, что и для вырожденных матриц.

● 8.2. Прямые методы решения систем линейных уравнений

Рассмотрим точные методы решения систем линейных уравнений. Метод решения называется *точным*, если за конечное число арифметических операций с точными числами можно получить точное решение системы уравнений. В противном случае метод *приближенный*. Заметим, что точные методы — это не настолько идеализированные алгоритмы, что на ЭВМ их нельзя реализовать. Для систем линейных уравнений с матрицей A и вектором b, состоящих из целых чисел, используя на ЭВМ арифметику целых чисел, можно с помощью точных методов находить точные решения.

8.2.1. Метод исключения Гаусса. К числу точных методов решения систем линейных уравнений относится широко применяемый в вычислительной практике метод Гаусса.

Рассмотрим систему уравнений (8.1.1). Алгоритм исключения состоит из последовательности шагов.

1-й шаг. Исключение неизвестной x_1 из всех уравнений системы (8.1.1), кроме первого. Пусть $a_{1,1} \neq 0$. Тогда из первого уравнения выражаем x_1

$$x_1 = \frac{(-1)}{a_{1,1}}\left(a_{1,2}x_2 + \ldots + a_{1,n}x_n\right) + b_1 \tag{8.2.1}$$

через остальные неизвестные и подставляем (8.2.1) во 2-е, 3-е, ..., n-е уравнение системы (8.1.1). После подстановки получаем преобразованную исходную систему вида

$$\begin{aligned} &a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n = b_1, \\ &a^{(1)}_{2,2}x_2 + \ldots + a^{(1)}_{2,n}x_n = b^{(1)}_2, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &a^{(1)}_{2,n}x_2 + \ldots + a^{(1)}_{n,n}x_n = b^{(1)}_n, \end{aligned} \tag{8.2.2}$$

где элементы матрицы $a^{(1)}_{ij}$, $2 \leqslant i, j \leqslant n$, и вектора $b^{(1)}_i$, $2 \leqslant i \leqslant n$, после первого шага получены по формулам

$$a^{(1)}_{i,j} = a_{i,j} - \frac{1}{a_{1,1}} a_{i,1}a_{1,j},$$

$$b^{(1)}_i = b_i - \frac{1}{a_{1,1}} a_{i,1}b_1.$$

2-й шаг. Исключение неизвестной x_2 из всех уравнений системы (8.2.2), кроме 1-го и 2-го. Пусть $a^{(1)}_{2,2} \neq 0$. Тогда из второго уравнения выражаем x_2

$$x_2 = \frac{(-1)}{a^{(1)}_{2,2}}\left(a^{(1)}_{2,3}x_3 + \ldots + a^{(1)}_{2,n}x_n\right) + b^{(1)}_2 \tag{8.2.3}$$

через остальные неизвестные и подставляем (8.2.3) в третье, четвертое, ..., n-е уравнения системы (8.2.2). После подстановки получаем преобразованную исходную систему

$$
\begin{aligned}
a_{1,1}x_1+\ldots \quad & +a_{1,n}x_n=b_1,\\
a^{(1)}_{2,2}x_2+\ldots \quad & +a^{(1)}_{2,n}x_n=b^{(1)}_2,\\
a^{(2)}_{3,3}x_3 \quad +\ldots & +a^{(2)}_{3,n}x_n=b^{(2)}_3,\\
\ldots\ldots\ldots\ldots\ldots & \ldots\ldots\ldots\ldots\ldots\\
a^{(2)}_{n,3}x_3 \quad +\ldots & +a^{(3)}_{n,n}x_n=b^{(2)}_n,
\end{aligned}
\tag{8.2.4}
$$

где элементы матрицы $a^{(2)}_{i,j}$, $3\leqslant i,j\leqslant n$, и вектора $b^{(2)}_i$, $3\leqslant i\leqslant n$, после второго шага получены по формулам

$$a^{(2)}_{i,j}=a^{(1)}_{i,j}-\frac{1}{a^{(1)}_{2,2}}a^{(1)}_{i,2}a^{(1)}_{2,j},$$

$$b^{(2)}_i=b_i-\frac{1}{a^{(1)}_{2,2}}a^{(1)}_{i,2}b^{(1)}_2.$$

Продолжая этот процесс исключения при условии, что

$$a^{(2)}_{3,3}\neq 0,\ \ldots,\ a^{(n-2)}_{n-1,n-1}\neq 0,$$

после $(n-1)$-го шага исключения получим преобразованную исходную систему

$$
\begin{aligned}
a_{1,1}x_1+\ldots \quad & +a_{1,n}x_n \quad =b_1,\\
a^{(1)}_{2,2}x_2+\ldots & +a^{(1)}_{2,n}x_n \quad =b^{(1)}_2,\\
\ldots\ldots\ldots\ldots & \ldots\ldots\ldots\ldots\ldots\\
& a^{(n-1)}_{n,n}x_n \quad =b^{(n-1)}_n
\end{aligned}
\tag{8.2.5}
$$

с верхней треугольной матрицей U

$$U=\begin{pmatrix} a_{1,1}a_{1,2}\ldots\ldots a_{1,n}\\ \quad a^{(1)}_{2,2} \quad\quad \\ \quad\quad \ddots \quad\quad \\ 0 \quad\quad\quad a^{(n-1)}_{n,n}\end{pmatrix}$$

и преобразованным вектором правых частей

$$g=\begin{pmatrix} b_1\\ b^{(1)}_2\\ \ldots\\ b^{(n-1)}_n\end{pmatrix}.$$

В матричной записи система уравнений (8.2.5) примет вид

$$Ux=g. \tag{8.2.6}$$

Преобразование исходной системы линейных уравнений к системе (8.2.6) с треугольной матрицей U — прямой ход исключения.

На этом этапе мы еще не вычислили ни одной компоненты вектора решения x, но эквивалентными преобразованиями привели систему к такой форме, для которой легко вычислить все компоненты решения x.

Пусть $a_{n,n}^{(n-1)} \neq 0$. Тогда осуществляем обратный ход: вычисляем компоненты вектора решения в обратном порядке. Из (8.2.5) находим

$$\begin{aligned} x_n &= \frac{b_n^{(n-1)}}{a_{n,n}^{(n-1)}}, \\ x_{n-1} &= \frac{1}{a_{n-1,n-1}^{(n-2)}}\left(b_{n-1}^{(n-2)} - a_{n-1,n}^{(n-2)} x_n\right), \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_1 &= \frac{1}{a_{1,1}}\left(b_1 - a_{1,n} x_n - \ldots - a_{1,2} x_2\right). \end{aligned} \tag{8.2.7}$$

Для примера решим систему уравнений

$$\begin{aligned} x_1 - 2x_2 + 3x_3 &= 1, \\ 2\,x_1 + 3x_2 + x_3 &= 2, \\ -x_1 - x_2 + x_3 &= 3. \end{aligned}$$

Первый шаг прямого хода исключения

$$\begin{aligned} x_1 - 2x_2 + 3x_3 &= 1, \\ 7x_2 - 7x_3 &= 0, \\ -3x_2 + 4x_3 &= 4; \end{aligned}$$

второй шаг

$$\begin{aligned} x_1 - 2x_2 + 3x_3 &= 1, \\ 7x_2 - 7x_3 &= 0, \\ x_3 &= 4; \end{aligned}$$

обратный ход:

$$\begin{aligned} x_3 &= 4, \\ x_2 &= 4, \\ x_1 &= -3. \end{aligned}$$

Определим нижнюю треугольную матрицу L с единичной главной диагональю формулой

$$\begin{pmatrix} 1 & & & & \\ \frac{a_{2,1}}{a_{1,1}} & 1 & & 0 & \\ \frac{a_{3,1}}{a_{1,1}} & \frac{a_{3,2}^{(1)}}{a_{2,2}^{(1)}} & 1 & & \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{a_{n,1}}{a_{1,1}} & \frac{a_{n,2}^{(1)}}{a_{2,2}^{(1)}} & \cdots & \frac{a_{n,n-1}^{(n-2)}}{a_{n-1,n-1}^{(n-2)}} & 1 \end{pmatrix}.$$

Прямой подставкой можно проверить, что справедливо разложение исходной матрицы A в произведение:

$$A = LU. \tag{8.2.8}$$

Представление матрицы A в виде (8.2.8) называется *LU-разложением матрицы.* Очевидно, что решение исходной системы уравнений эквивалентно решению двух систем с треугольными матрицами

$$Lg = b, \tag{8.2.9}$$

$$Ux = g. \tag{8.2.10}$$

Таким образом, рассмотренный вариант метода исключения Гаусса — это определение вектора g решением (8.2.9) и затем определение вектора x решением (8.2.10).

Справедливо следующее утверждение.

Теорема 8.4. *Для существования LU-разложения матрицы A необходимо и достаточно, чтобы у матрицы A все главные миноры были отличны от нуля.*

У произвольной невырожденной матрицы A главные миноры, т. е.

$$\begin{vmatrix} a_{1,1} \end{vmatrix}, \quad \begin{vmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{vmatrix}, \quad \ldots \quad \begin{vmatrix} a_{1,1} & \ldots & a_{1,n} \\ \cdots & \cdots & \cdots \\ a_{n,1} & \ldots & a_{n,n} \end{vmatrix},$$

вообще говоря, могут обращаться в нуль. Например, матрица

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

невырожденная, а ее первый главный минор равен нулю.

Чтобы применить метод Гаусса к таким матрицам, или, что то же самое, получить LU-разложение, необходимо произвести перестановку строк (или столбцов) так, чтобы главные миноры стали отличными от нуля.

Теорема 8.5. *Произвольная невырожденная матрица перестановкой строк (столбцов) может быть приведена к матрице с главными минорами, отличными от нуля.*

Обычно перестановку строк (столбцов) не производят отдельно от процедуры исключения, а соединяют эти два процесса в один.

Если $a_{1,1}=0$, переставим строки матрицы A так, чтобы в левом верхнем углу оказался ненулевой элемент. В первом столбце такой элемент всегда найдется, иначе $\det A=0$. Если после первого шага получим $a_{2,2}^{(1)}=0$, то выполним, как и выше, перестановку; во втором столбце всегда найдется ненулевой элемент, иначе два первых столбца были бы линейно зависимы и $\det A=0$. Поместим строку с ненулевым элементом во втором столбце на место второй строки, тогда $a_{2,2}^{(1)}\neq 0$. Продолжая этот процесс исключения и перестановки строк (если элемент $a_{k,k}^{(k-1)}=0$) до $k=n$, получим LU-разложение матрицы A с дополнительной матрицей P-матрицей перестановок строк:

$$PA=LU. \tag{8.2.11}$$

Матрица P получается из единичной матрицы E перестановкой тех же строк. Например, перестановке 2-й и 4-й строк матрицы соответствует

$$P=\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Заметим, что матрица P — это и есть та матрица перестановок, существование которой утверждается в теореме 8.5.

8.2.2. Вычисление определителя и обратной матрицы с помощью LU-разложения. Напомним формулу для определителя произведения двух матриц:

$$\det A=\det L\,\det U.$$

Заметим, что определитель треугольных матриц равен произведению диагональных элементов. Следовательно,

$$\det L=1,$$
$$\det U=a_{1,1}\cdot a_{2,2}^{(1)}\ldots a_{n,n}^{(n-1)}.$$

Отсюда значение определителя матрицы A вычисляется по формуле

$$\det A=a_{1,1}\cdot a_{2,2}^{(1)}\cdot\ldots\cdot a_{n,n}^{(n-1)}.$$

Если матрица A представлена в форме (8.2.11) с перестановками, то

$$\det P\det A=\det L\det U.$$

Но так как матрица P получается перестановкой строк единичной матрицы E, то

$$\det P=\begin{cases} 1 \text{ при четном числе } k \text{ перестановок,} \\ -1 \text{ при нечетном числе } k \text{ перестановок.} \end{cases}$$

Окончательно определитель матрицы A равен

$$\det A=(-1)^k a_{1,1}\cdot a_{2,2}^{(1)}\ldots a_{n,n}^{(n-1)}.$$

Напомним формулу для обратной матрицы, представленной произведением двух матриц:

$$A=LU,\ A^{-1}=U^{-1}L^{-1}.$$

Матрицы L, U — треугольные, поэтому вычисление элементов L^{-1} и U^{-1} не представляет труда (см. (8.2.7)). Если матрица A представлена в форме (8.2.11) с перестановками, то

$$A^{-1}P^{-1}=U^{-1}L^{-1},\ A^{-1}=U^{-1}L^{-1}P,$$

матрица P^{-1} есть матрица обратной перестановки, возвращающей строки на их прежние места. Можно заметить, что

$$P^{-1}=P',$$

где P' — транспонированная к P матрица.

В реальных вычислениях обратная матрица A^{-1} находится решением n систем линейных уравнений методом исключения Гаусса:

$$Ax=e_i,\ 1\leqslant i\leqslant n,$$

где e_i — i-столбец единичной матрицы E. Полученные векторы решений x_1, x_2, ..., x_n образуют n столбцов матрицы A^{-1}, поскольку $AA^{-1}=E$.

8.2.3. Разложение на треугольные множители симметричной положительно определенной матрицы. Симметричные положительно определенные матрицы A очень часто связаны с изучением поведения механических систем в окрестности устойчивого положения равновесия. Матрица A трактуется тогда как матрица квадратичной формы в записи энергии W:

$$W=(Ax,\ x)=\sum_{i,j=1}^{n} a_{i,j}x_i x_j,$$

где x_i — обобщенные координаты.

Для положительно определенных матриц выполняются детерминантные неравенства Сильвестра

$$a_{1,1}>0,\quad \begin{vmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{vmatrix}>0,\ \ldots,\ \begin{vmatrix} a_{1,1} & \ldots & a_{1,n} \\ \ldots & \ldots & \ldots \\ a_{n,1} & \ldots & a_{n,n} \end{vmatrix}>0,$$

откуда в силу теоремы 8.4 сразу же следует существование LU-разложения.

Однако для рассматриваемого класса матриц, если не требовать, чтобы на диагонали L стояли единицы, справедливо следующее разложение:

$$A=LL', \tag{8.2.12}$$

где L' — матрица, транспонированная к L. Действительно, определим элементы матрицы L, сравнивая каждый элемент матрицы в левой и правой части (8.2.12). Имеем

$$\begin{pmatrix} a_{1,1} \dots a_{1,n} \\ \vdots \quad \vdots \\ a_{n,1} \dots a_{n,n} \end{pmatrix} = \begin{pmatrix} l_{1,1} & & 0 \\ \vdots & \ddots & \\ l_{1,n} & \dots & l_{n,n} \end{pmatrix} \begin{pmatrix} l_{1,1} \dots l_{1,n} \\ & \ddots \quad \vdots \\ 0 & \quad l_{n,n} \end{pmatrix},$$

откуда получаем равенства для первого столбца

$$l_{1,1}=(a_{1,1})^{1/2},\ l_{i,1}=(a_{i,1})/l_{1,1},\ 2\leqslant i\leqslant n.$$

Аналогично находим соотношения

$$a_{i,i}=\sum_{k=1}^{i} l_{i,k}^2,\ a_{i,j}=\sum_{k=1}^{j} l_{i,k}l_{j,k},\ j<i,$$

которые приводят к формулам

$$l_{i,i}=\left(a_{i,i}-\sum_{k=1}^{i-1} l_{i,k}^2\right)^{1/2},\quad 1\leqslant i\leqslant n,$$

$$l_{i,j}=\left(a_{i,j}-\sum_{k=1}^{i-1} l_{i,k}l_{j,k}\right)/l_{i,i},\quad j+1\leqslant i\leqslant n.$$

После того как выполнено разложение (8.2.12), решение системы линейных уравнений $Ax=b$ получается решением двух систем с треугольными матрицами:

$$Lg=b,$$
$$L'x=g.$$

В отличие от случая общих матриц изложенный метод разложения требует меньше памяти ЭВМ, так как хранится только треугольная часть матрицы A и не нужна перестановка строк.

8.2.4. Вычислительная погрешность метода Гаусса и выбор ведущего элемента исключения. Выше отмечалось, что метод исключения является точным методом. При этом предполагалось, что арифметические операции выполняются над точными числами. Если же метод исключения реализуется на ЭВМ, то появляется вычислительная погрешность, связанная с конечным числом разрядов ЭВМ для представления вещественных чисел, ошибками округления, реализацией на ЭВМ арифметических операций.

Вычислительная погрешность метода исключения может быть устранена в арифметике целых чисел. При этом если деление на элементы

$$a_{1,1},\ a_{2,2}^{(1)},\ \dots,\ a_{n,n}^{(n-1)} \tag{8.2.13}$$

производится с остатком, то оно не выполняется, а запоминаются числитель и знаменатель. Компоненты вектора решения определяются в виде дробей. Метод Гаусса в арифметике целых чисел для общих матриц A требует определенного искусства масштабирования, чтобы избежать переполнения после арифметических операций.

Пусть метод исключения реализуется в арифметике вещественных чисел. Тогда естественно поставить вопрос об уменьшении вычислительной погрешности за счет модификации алгоритма исключения. Фактически идея такой модификации уже была использована выше. Вводились матрицы перестановок P, чтобы избежать деления на нуль. Деление привело бы к бесконечно большой вычислительной погрешности.

Элементы (8.2.13), на которые производится деление в методе Гаусса, называются *ведущими элементами исключения*.

Уменьшение вычислительной погрешности производится путем перестановки строк и столбцов матрицы так, чтобы ведущий элемент на k-м шаге исключения был наибольшим по модулю из коэффициентов, участвующих в дальнейшем исключении:

$$|a_{k,k}^{(k-1)}| = \max_{k \leqslant i,j \leqslant n} |a_{i,j}^{(k-1)}|, \quad 1 \leqslant k \leqslant n, \tag{8.2.14}$$

где $a_{i,j}^{(0)} = a_{i,j}$.

Выбор ведущих элементов по формуле (8.2.14) называют *методом исключения с полным упорядочением*.

Полное упорядочение требует большой дополнительной вычислительной работы, поэтому часто останавливаются на методе исключения с частичным упорядочением по строкам. Выбор ведущих элементов производится согласно формуле

$$|a_{k,k}^{(k-1)}| = \max_{k \leqslant i \leqslant n} |a_{i,k}^{(k-1)}|. \tag{8.2.15}$$

Например, для системы уравнений

$$\begin{aligned} 4x_1 + 5x_2 + 12x_3 &= 1, \\ x_1 - 3x_2 + x_3 &= 2, \\ 10x_1 + x_2 - x_3 &= 3 \end{aligned}$$

перед первым шагом исключения требуется согласно алгоритму с полным упорядочением поменять первый и третий столбцы системы уравнений:

$$\begin{aligned} 12x_3 + 4x_1 + 5x_2 &= 1, \\ x_3 + x_1 - 3x_2 &= 2, \\ -x_3 + 10x_1 + x_2 &= 3; \end{aligned}$$

согласно алгоритму с частичным упорядочением поменять первую и третью строки системы уравнений:

$$\begin{aligned} 10x_1 + x_2 - x_3 &= 3, \\ x_1 - 3x_2 + x_3 &= 2, \\ 4x_1 + 5x_2 + 12x_3 &= 1. \end{aligned}$$

Затем выполняется 1-й шаг исключения.

Решая конкретную систему уравнений, трудно, вообще говоря, априори оценить по матрице A и вектору погрешность как

системы уравнений (найти число обусловленности), так и вычислительную погрешность. Поэтому можно предложить следующие практические рекомендации:

1) для оценки числа обусловленности следует численно решить несколько систем уравнений с близкими правыми частями к заданному вектору b;

2) для оценки вычислительной погрешности следует решить систему уравнений с двойной точностью.

8.2.5. Оценка числа арифметических операций метода Гаусса. Процесс исключения Гаусса запишем на фортране, используя обозначения $l_{i,k}=\mathrm{L(I,K)}$, $a_{i,j}=\mathrm{A(I,J)}$, $b_i=\mathrm{B(I)}$, $x_i=\mathrm{X(I)}$, $n=\mathrm{N}$, массив L должен быть описан как вещественный. Имеем следующую программу

```
C       ПРЯМОЙ ХОД
        DO  1  K=1,N-1
        DO  1  I=K+1,N
        L(I,K)=A(I,K)/A(K,K)
        DO  10 J=K+1,N
10      A(I,J)=A(I,J)-L(I,K)*A(K,J)
1       B(I)=B(I)-L(I,K)*B(K)
C       ОБРАТНЫЙ ХОД
        DO  3  K=N,1,-1
        X(K)=0.
        DO  2  J=K+1,N
2       X(J)=X(J)+A(K,J)*X(J)
3       X(K)=(B(K)-X(K))/A(K,K)
```

Здесь элементы матрицы $a_{i,j}^{(k)}$ записываются в память на место исходной матрицы $a_{i,j}$, элементы $b_i^{(k)}$ — на место вектора b_i (см. формулы (8.2.5), (8.2.7)).

Обратившись к тексту программы, замечаем, что при больших значениях порядка системы уравнений n основная вычислительная работа сосредоточена в трех вложенных циклах DO 1. Число арифметических операций при $n\to\infty$ есть $O(n^3)$, и основное время затрачивается на приведение матрицы к треугольному виду.

Для примера число арифметических операций решения системы уравнений с матрицей размером 100×100 (т. е. $n=100$) имеет порядок 10^6. На ЭВМ с быстродействием 10^5 оп/с время решения такой системы будет порядка 10 с.

Если приведенный выше алгоритм дополнить выбором ведущего элемента исключения с частичным упорядочением, то между циклом по K и циклом по I появится еще один цикл

```
        DO 1  K=1,N-1
        DO 11 M=K,N
        .....
11      CONTINUE
        DO 1  I=K+1, N
        .....
```

В этом цикле производится выбор ведущего элемента A(K, K), что потребует $O(n)$ арифметических операций (вычитания). Общее число арифметических операций при $n \to \infty$ подсчитаем следующим образом: в циклах по I, J операций $O(n^2)$ плюс в цикле по M: $O(n)$, откуда суммарное число операций

$$O(n)(O(n^2)+O(n)^2)=O(n^3)$$

остается того же порядка, что и в алгоритме без упорядочения и основное время затрачивается на приведение матрицы к треугольному виду.

Наконец, легко понять, что при полном упорядочении затраты на выбор ведущего элемента исключения сравниваются по порядку величины с приведением матрицы к треугольному виду.

8.2.6. Применение библиотечных программ. Вычисление определителя вещественной матрицы A с помощью программы A8A5 выполняется следующим образом. Для примера пусть

$$A=\begin{pmatrix}1 & 3\\ 2 & 4\end{pmatrix}.$$

Тогда программа может иметь вид

```
      REAL  A(2,2),D,W(2)
      INTEGER  K,N,I
      DATA  A/1.,2.,3.,4./,K,N,I/2,2,0/
C     ОБРАЩЕНИЕ К ПРОГРАММЕ A8A5
      CALL  A8A5(A,K,N,D,W,I)
C     ВЫВОД НА ТЕРМИНАЛ
      WRITE (5,1) D,I
1     FORMAT (2X,'D=',E13.6,'I=',I2)
      END
```

Решение системы линейных уравнений с симметричной положительно определенной матрицей

и набором правых частей

$$b_1=\begin{pmatrix}1\\ 2\end{pmatrix};\quad b_2=\begin{pmatrix}3\\ 4\end{pmatrix};\quad b_3=\begin{pmatrix}5\\ 6\end{pmatrix}$$

с помощью программы A9A0 выполняется следующим образом:

```
      REAL  A(2,2),B(2,3),W(2),X(2,3),W(2,3)
      INTEGER  K,L,N,M,J,L,I
      DATA  A/3.,1.,1.,4./,B/1.,2.,3.,4.,5.,6./
      DATA  K,L,N,M,J,L1,I/2,2,2,3,2,2,0/
C     ОБРАЩЕНИЕ К ПРОГРАММЕ A9A0
      CALL  A9A0(A,K,B,L,N,M,X,J,W,W1,L1,I)
C     ВЫВОД НА ТЕРМИНАЛ
```

```
      WRITE (5,1) X,I
1     FORMAT (2X,'X=',/,6E3.6,/,'I=',I2)
      END
```

● 8.3. Итерационные методы решения систем линейных уравнений

8.3.1. Введение. Рассмотрим итерационные методы решения систем линейных уравнений. Эти методы, вообще говоря, не дают возможности найти точное решение за конечное число итераций даже при отсутствии вычислительной погрешности. С их помощью строится последовательность $\{x^{(k)}\}$ такая, что

$$\lim_{k\to\infty} x^{(k)} = x,$$

где x — точное решение.

Область широкого применения итерационных методов — это системы уравнений, к которым приводят численные методы решения дифференциальных уравнений с частными производными. Матрицы таких систем имеют большое число нулевых элементов, и в отличие от прямых методов итерационные не увеличивают число ненулевых элементов матрицы в процессе вычислений. Эффективность итерационных методов определяется скоростью сходимости последовательных приближений $x^{(k)}$ к решению.

Пусть ищется решение невырожденной системы уравнений

$$Ax = b. \tag{8.3.1}$$

Первым шагом в итерационном методе является преобразование исходной системы к виду

$$Cx = Bx + d, \tag{8.3.2}$$

где матрицы C, B и вектор d определяются по матрице A и вектору b. Причем системы (8.3.1) и (8.3.2) являются эквивалентными, т. е. их решения совпадают, а построение обратной матрицы C^{-1} проще, чем A^{-1}.

Вторым шагом является расстановка индексов или номеров приближений в (8.3.2) и задание нулевого приближения. Например,

$$Cx^{(k+1)} = Bx^{(k)} + d, \quad k = 0, 1, 2, \ldots, \tag{8.3.3}$$

где $x^{(0)}$ — заданный вектор $x^{(0)} = (x_1^{(0)}, \ldots, x_n^{(0)})$.

Третьим шагом итерационного метода является обоснование сходимости последовательных приближений $\{x^{(k)}\}$, полученных из (8.3.3), к точному решению x системы (8.3.1) и оценка погрешности k-го приближения

$$\| x^{(k)} - x \| \leqslant \varepsilon. \tag{8.3.4}$$

Оценка (8.3.4) при заданном ε позволяет остановить итерационный процесс (8.3.3).

Различные итерационные методы отличаются первыми двумя шагами, а выбор конкретного метода должен производиться на основании оценки (8.3.4).

8.3.2. Метод простой итерации. В методе простой итерации матрица C (8.3.2) выбирается единичной:

$$C=E.$$

Итерационный процесс описывается формулой

$$x^{(k+1)}=Bx^{(k)}+d,\quad k=0,\ 1,\ \ldots, \tag{8.3.5}$$

где $x^{(0)}$ — заданный вектор. В координатной записи (8.3.5) имеет вид

$$\begin{aligned}
x_1^{(k+1)}&=b_{1,1}x_1^{(k)}+\ldots+b_{1,n}x_n^{(k)}+d_1,\\
x_2^{(k+1)}&=b_{2,1}x_1^{(k)}+\ldots+b_{2,n}x_n^{(k)}+d_2,\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
x_n^{(k+1)}&=b_{n,1}x_1^{(k)}+\ldots+b_{n,n}x_n^{(k)}+d_n.
\end{aligned}$$

Теорема 8.6. *Для сходимости последовательных приближений* $\{x^{(k)}\}$ (8.3.5) *к точному решению* x *системы уравнений*

$$x=Bx+d \tag{8.3.6}$$

достаточно, чтобы

$$\|B\|<1. \tag{8.3.7}$$

Доказательство. Пусть $x^{(0)}$ — произвольный вектор. Из (8.3.5) находим

$$\begin{aligned}
x^{(1)}&=Bx^{(0)}+d,\\
x^{(2)}&=Bx^{(1)}+d=B\left(Bx^{(0)}+d\right)+d=B^2x^{(0)}+Bd+d,\\
x^{(3)}&=Bx^{(2)}+d=B\left(B^2x^0+Bd+d\right)+d=B^3x^{(0)}+\left(B^2+B+E\right)d,\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
x^{(k+1)}&=B^{k+1}x^{(0)}+\left(B^k+B^{k-1}+\ldots+E\right)d.
\end{aligned} \tag{8.3.8}$$

Покажем, что последовательность $\{x^{(k)}\}$ имеет предел, если выполнено условие (8.3.7). Действительно, в правой части (8.3.8) первое слагаемое имеет своим пределом нулевой вектор, поскольку

$$\|B^{k+1}x^{(0)}\|\leqslant\|B\|^{k+1}\|x^{(0)}\|\xrightarrow[k\to\infty]{}0.$$

Второе слагаемое в (8.3.8) имеет предел, если сходится матричный ряд $\sum\limits_{i=0}^{\infty}B^i$. Но из (8.3.7) следует, что собственные числа матрицы B удовлетворяют неравенству (см. теорему 5.2) $|\lambda(B)|<1$. Тогда из теоремы 5.1 последует сходимость матричного ряда к сумме — матрице

$$(E-B)^{-1}=\sum_{i=0}^{\infty}B^i.$$

Переходя в (8.3.8) к пределу при $k\to\infty$, получаем

$$x=(E-B)^{-1}d,$$

а это есть решение (8.3.6), что и требовалось доказать.

Для иллюстрации итерационного процесса рассмотрим систему уравнений

$$x_1 = 0{,}1x_1 - 0{,}2x_2 + 1,$$
$$x_2 = 0{,}05x_1 + 0{,}1x_2 - 1.$$

Здесь

$$\| B \| = \max_{1 \leqslant i \leqslant 2} \sum_{j=1}^{2} |b_{i,j}| = 0{,}3 < 1.$$

Итерационный процесс

$$x_1^{(k+1)} = 0{,}1x_1^{(k)} - 0{,}2x_2^{(k)} + 1,$$
$$x_2^{(k+1)} = 0{,}05x_1^{(k)} + 0{,}1x_2^{(k)} - 1$$

с начальным вектором $x_1^{(0)} = 0$, $x_2^{(0)} = 0$ дает следующие приближения:

$$x_1^{(1)} = 1{,}0; \quad x_1^{(2)} = 0{,}1 \cdot 1{,}0 - 0{,}2(-1,\ 0) + 1{,}0 = 1{,}3; \quad x_1^{(3)} = \ldots;$$
$$x_2^{(1)} = -1{,}0; \quad x_2^{(2)} = 0{,}05 \cdot 1{,}0 + 1{,}0(-1,\ 0) - 1{,}0 = -1{,}05; \quad x_2^{(3)} = \ldots .$$

Оценим число арифметических операций, выполняемых за один шаг итерационного процесса. Запишем (8.3.5) на фортране, используя следующие обозначения: вектор $x^{(k+1)} \to$ X1 (I), $d \to$ D (I), $x^{(k)} \to$ X (I), матрица $B \to$ B (I,J), $1 \leqslant$ I, J $\leqslant$ N. Имеем

```
      DO 2 I=1,N
      S=0.
      DO 1 J=1,N
1     S=S+B(I,J)*X(J)
2     X(I)=S+D(I)
```

Таким образом, при $n \to \infty$ число арифметических операций $O(n^2)$. Если заданная точность вычислений ε достигается за K итераций, то общее число операций $O(Kn^2)$. Сравнивая с вычислительными затратами метода исключения $O(n^3)$, можно сделать вывод, что при больших n метод простой итерации будет менее трудоемким.

Слабым местом метода простой итерации, да и других итерационных методов, является отсутствие единого практического подхода к поиску преобразования (8.3.1) в систему (8.3.6), чтобы выполнялось условие сходимости (8.3.7) для любых невырожденных матриц A, хотя для некоторых частных видов A такое преобразование и можно указать, например для матриц A с преобладанием диагональных элементов

$$|a_{i,i}| > \sum_{\substack{j=1 \\ i \neq j}}^{n} |a_{i,j}|, \quad 1 \leqslant i \leqslant n.$$

Каждое из уравнений системы (8.3.1)

$$a_{i,1}x_1 + \ldots + a_{i,n}x_n = b_i, \quad 1 \leqslant i \leqslant n,$$

разделим на $a_{i,i}$ и перенесем члены с $x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$ в правую часть. Получим

$$x_i = -\frac{a_{i,1}}{a_{i,i}}x_1 - \ldots - \frac{a_{i,i-1}}{a_{i,i}}x_{i-1} - \frac{a_{i,i+1}}{a_{i,i}}x_{i+1} - \ldots - \frac{a_{i,n}}{a_{i,i}}x_n + \frac{b_i}{a_{i,i}}.$$

Нетрудно проверить, что матрица правой части B имеет норму меньше 1. В обозначениях

$$b_{i,j}=-\frac{a_{i,j}}{a_{i,i}},\ i\neq j,\ b_{i,i}=0;\ d_i=\frac{b_i}{a_{i,i}}$$

итерационный процесс (8.3.5) сходится.

8.3.3. Оценка погрешности метода простой итерации. Возьмем в качестве вектора $x^{(0)}$ нулевой вектор.

Теорема 8.7. *Пусть* $\|B\|<1$. *Тогда имеет место неравенство*

$$\|x^{(k)}-x\|\leqslant\frac{\|B\|^k}{1-\|B\|}\|d\|,\ k\geqslant 1, \tag{8.3.9}$$

где $x^{(k)}$ — k-е *приближение, полученное из* (8.3.5), x — *точное решение* (8.3.6).

Доказательство. Приближение $x^{(k)}$ определяется из (8.3.8)

$$x^{(k)}=\left(B^{k-1}+B^{k-2}+\ldots+E\right)d.$$

Точное решение:

$$x=\left(B^{k-1}+B^{k-2}+\ldots+E\right)d+\left(B^k+B^{k+1}+\ldots\right)d.$$

Вычитая из первого равенства второе и оценивая нормы, получим

$$\|x^{(k)}-x\|=\|B^k+B^{k+1}+\ldots\|\,\|d\|\leqslant\left(\|B\|^k+\|B\|^{k+1}+\ldots\right)\|d\|. \tag{8.3.10}$$

Сумма $\left(\|B\|^k+\|B\|^{k+1}+\ldots\right)$ — это сумма бесконечной геометрической прогрессии со знаменателем $\|B\|$; она равна

$$\|B\|^k/(1-\|B\|).$$

Отсюда и из (8.3.10) следует (8.3.9), что и требовалось доказать.

Оценка (8.3.9) называется *априорной оценкой погрешности*, так как, не проводя вычислений, по $\|B\|$ и $\|d\|$ можно оценить погрешность k-го приближения $x^{(k)}$, а по заданной точности ε легко определяется необходимое число итераций. Действительно,

$$\|B\|^k\,\|d\|=\varepsilon\left(1-\|B\|\right),$$

$$K=\frac{1}{\ln\left(\|B\|\right)}\left(\ln\left(\varepsilon\left(1-\|B\|\right)/\|d\|\right)\right).$$

Теорема 8.8. *Пусть* $\|B\|<1$. *Тогда имеет место неравенство*

$$\|x^{(k)}-x\|\leqslant\frac{\|B\|}{1-\|B\|}\|x^{(k)}-x^{(k-1)}\|. \tag{8.3.11}$$

Доказательство. Заметим, что

$$x^{(k)}-x=B\left(x^{(k-1)}-x\right). \tag{8.3.12}$$

Прибавим к левой и правой части этого равенства $x^{(k-1)}$ и перепишем его следующим образом:

$$x^{(k-1)}-x=x^{(k-1)}-x^{(k)}+B\left(x^{(k-1)}-x\right).$$

Приравняв нормы левой и правой части и воспользовавшись неравенством треугольника, получим

$$\|x^{(k-1)}-x\|\leqslant\|x^{(k-1)}-x^{(k)}\|+\|B\|\,\|x^{(k-1)}-x\|.$$

Отсюда следует неравенство

$$\| x^{(k-1)} - x \| \leqslant \frac{\| x^{(k)} - x^{(k-1)} \|}{1 - \| B \|}.$$

Из (8.3.12) имеем

$$\| x^{(k)} - x \| \leqslant \| B \| \, \| x^{(k-1)} - x \|.$$

Последние два неравенства дают оценку (8.3.11), что и требовалось доказать.

Оценка (8.3.11) называется *апостериорной оценкой погрешности*, так как ее получают по известной $\| B \|$ и вычисленным приближениям $x^{(k-1)}$ и $x^{(k)}$. Целесообразно получать как априорную, так и апостериорную оценку погрешности, поскольку их сравнение дает возможность оценить вычислительную погрешность итерационного процесса.

8.3.4. Метод Зейделя. Отличие метода Зейделя от простой итерации состоит лишь в том, что при вычислении $(k+1)$-го приближения полученные $x^{(k+1)}$ компоненты вектора $x^{(k+1)}$ сразу же используются в вычислениях. В координатной записи итерационный процесс Зейделя имеет вид

$$\begin{aligned}
x_1^{(k+1)} &= b_{1,1} x_1^{(k)} + b_{1,2} x_2^{(k)} + \ldots + b_{1,n} x_n^{(k)} + d_1, \\
x_2^{(k+1)} &= b_{2,1} x_1^{(k+1)} + b_{2,2} x_2^{(k)} + \ldots + b_{2,n} x_n^{(k)} + d_2, \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
x_n^{(k+1)} &= b_{n,1} x_1^{(k+1)} + b_{n,2} x_2^{(k+1)} + \ldots + b_{n,n} x_n^{(k)} + d_n,
\end{aligned} \tag{8.3.13}$$

начальный вектор $x^{(0)} = (x_1^{(0)}, \ldots, x_n^{(0)})$ задается; $k = 0, 1, 2, \ldots$. В матричной записи (8.3.13) можно представить так:

$$x^{(k+1)} = U x^{(k)} + L x^{(k+1)} + d,$$

где матрицы U, L получены разложением B в сумму:

$$B = U + L;$$

матрица U — верхняя треугольная часть B, включая диагональ; L — нижняя поддиагональная часть B;

$$U = \begin{pmatrix} b_{1,1} & \ldots & b_{1,n} \\ & \ddots & \vdots \\ 0 & & b_{n,n} \end{pmatrix}; \quad L = \begin{pmatrix} 0 & \ldots & 0 \\ b_{2,1} & \ddots & \vdots \\ \ldots & \ddots & \ddots \\ b_{n,1} & \ldots & b_{n,n-1} \; 0 \end{pmatrix}.$$

Таким образом, метод Зейделя можно записать в форме (8.3.3), если принять $C = (E - L)$, $B = U$. Имеем

$$(E - L) x^{(k+1)} = U x^{(k)} + d, \quad k = 0, 1, 2, \ldots . \tag{8.3.14}$$

Заметим, что построение матрицы, обратной $(E - L)^{-1}$, не представляет труда, так как это нижняя треугольная матрица.

Теорема 8.9. *Пусть выполнено неравенство*

$$\|(E-L)^{-1}U\|<1; \tag{8.3.15}$$

тогда последовательные приближения $\{x^{(k)}\}$, *полученные из* (8.3.14), *сходятся к точному решению системы*

$$x=Bx+d$$

при любом начальном векторе $x^{(0)}$.

Доказательство. Заметим, что метод Зейделя (8.3.14) — это метод простой итерации для системы

$$x=(E-L)^{-1}Ux+(E-L)^{-1}d,$$

которая эквивалентна исходной $x=Bx+d$. Обозначая $(E-L)^{-1}U=B_1$, $(E-L)^{-1}d=d_1$, запишем (8.3.14) следующим образом:

$$x^{(k+1)}=B_1x^{(k)}+d_1.$$

Теперь утверждение теоремы следует из теоремы 8.6.

Заметим, что оценка погрешности k-го приближения метода Зейделя в случае выполнения (8.3.15) вытекает из (8.3.9), (8.3.11) при условии замены B на B_1:

$$\|x^{(k)}-x\|\leqslant\frac{\|B_1\|^k\|d\|}{1-\|B_1\|};\quad \|x^{(k)}-x\|\leqslant\frac{\|B_1\|}{1-\|B_1\|}\|x^{(k)}-x^{(k-1)}\|.$$

Выше приводились достаточные условия сходимости последовательных приближений ($\|B\|<1$, $\|B_1\|<1$). Заметим, что области сходимости метода простой итерации и Зейделя лишь пересекаются: существуют матрицы B, для которых метод Зейделя сходится, а метод простой итерации расходится, и наоборот.

Для иллюстрации итерационного процесса Зейделя рассмотрим систему

$$\begin{cases}x_1=0{,}1x_1-0{,}2x_2+1,\\ x_2=0{,}5x_1+0{,}1x_2-1.\end{cases}$$

Здесь

$$B=U+L;$$

$$B=\begin{pmatrix}0{,}1 & -0{,}2\\ 0{,}05 & 0{,}1\end{pmatrix},\quad U=\begin{pmatrix}0{,}1 & -0{,}2\\ 0 & 0{,}1\end{pmatrix},\quad L=\begin{pmatrix}0 & 0\\ 0{,}05 & 0\end{pmatrix},$$

$$E-L=\begin{pmatrix}1 & 0\\ -0{,}05 & 1\end{pmatrix},\quad (E-L)^{-1}=\begin{pmatrix}1 & 0\\ 0{,}05 & 1\end{pmatrix},$$

$$(E-L)^{-1}U=\begin{pmatrix}0{,}1 & -0{,}2\\ 0{,}005 & 0{,}09\end{pmatrix},\quad \|B_1\|=0{,}3<1.$$

Итерационный процесс

$$\begin{aligned}x_1^{(k+1)}&=0{,}1x_1^{(k)}-0{,}2x_2^{(k)}+1,\\ x_2^{(k+1)}&=0{,}05x_1^{(k+1)}+0{,}1x_2^{(k)}-1\end{aligned}$$

с начальным вектором $x_1^{(0)}=0$, $x_2^{(0)}=0$ дает следующие приближения:

$$x_1^{(1)}=1{,}00; \quad x_1^{(2)}=1{,}2900; \quad x_1^{(3)}=\ldots$$
$$x_2^{(1)}=-0{,}95; \quad x_2^{(2)}=-1{,}0305; \quad x_2^{(3)}=\ldots$$

● 8.4. Вычисление собственных значений и векторов

8.4.1. Введение. Большинство алгоритмов определения собственных значений и векторов матрицы A основаны на приведении A с помощью невырожденных преобразований к такой матрице A^*, для которой проблема собственных значений является простой. Например, если A — вещественная матрица с различными вещественными собственными значениями $\lambda_1, \ldots, \lambda_n$, то матрица преобразования T, которая приводит A к диагональному виду, полностью решает проблему. Действительно,

$$A^*=\begin{pmatrix} a_{1,1}^* & & 0 \\ & \ddots & \\ 0 & & a_{n,n}^* \end{pmatrix}=T^{-1}AT.$$

Столбцы матрицы T образуют полную систему собственных векторов матрицы A, а диагональные элементы A^* суть собственные значения A.

Чтобы определить собственные значения матрицы (при тех же предположениях относительно λ_i), не обязательно приводить ее к диагональному виду. Достаточно привести ее к треугольному виду, поскольку на диагонали будут расположены собственные значения матрицы A:

$$A^*=\begin{pmatrix} a_{1,1}^* & \ldots & a_{1,n}^* \\ & \ddots & \\ & & \ddots \\ 0 & & a_{n,n}^* \end{pmatrix}=T^{-1}AT.$$

Действительно, характеристическое уравнение A^*

$$\det\left(A^*-\lambda E\right)=0$$

представляется в форме

$$\left(a_{1,1}^*-\lambda\right)\left(a_{2,2}^*-\lambda\right)\ldots\left(a_{n,n}^*-\lambda\right)=0,$$

откуда следует, что собственные значения A^*

$$\lambda\left(A^*\right)=\{a_{1,1}^*, a_{2,2}^*, \ldots, a_{n,n}^*\}.$$

Остается показать, что $\lambda(A)=\lambda\left(A^*\right)$. Это вытекает из равенств

$$\det\left(A^*-\lambda E\right)=\det\left(T^{-1}AT-\lambda E\right)=\det\left(T^{-1}\left(A-\lambda E\right)T\right)=$$
$$=\det T^{-1}\det\left(A-\lambda E\right)\det T=\det\left(A-\lambda E\right).$$

Определение собственных векторов e_i^* треугольной матрицы A^* является простой задачей, так как она эквивалентна решению

системы однородных уравнений $A^*e_i = a^*_{i,i}e_i$, $1 \leqslant i \leqslant n$, с треугольной матрицей и каким-либо дополнительным условием, однозначно выделяющим собственные векторы, например $e^*_{i,i} = 1$, $1 \leqslant i \leqslant n$. Зная собственные векторы матрицы A^*, можно получить собственные векторы A с помощью матрицы преобразования T

$$e_i = Te^*_i,$$

поскольку из равенства

$$T^{-1}ATe^*_i = a^*_{i,i}e^*_i$$

следует соотношение

$$A(Te^*_i) = a^*_{i,i}(Te^*_i).$$

Заметим, что вещественная матрица, если она имеет комплексные собственные значения, не может быть приведена к треугольной матрице A^* с помощью вещественного преобразования. Матрицей, к которой можно привести любую матрицу A с помощью вещественного и даже ортогонального преобразования, является почти треугольная матрица A^*:

$$A^* = \begin{pmatrix} a^*_{1,1} & \cdots & a^*_{1,n} \\ a^*_{2,1} & \cdots & a^*_{2,n} \\ & \ddots & \\ 0 & & a^*_{n,n-1}\, a^*_{n,n} \end{pmatrix}. \qquad (8.4.1)$$

В задаче определения собственных значений и собственных векторов первым шагом является приведение матрицы A к форме (8.4.1). Эту процедуру целесообразно выполнить для того, чтобы дальнейшие вычисления вести с ненулевыми элементами A^*, которых при больших n почти вдвое меньше, чем у исходной матрицы A. Для симметричных матриц A форма (8.4.1) принимает вид

$$A^* = \begin{pmatrix} a^*_{1,1}\, a^*_{1,2} & & & 0 \\ a^*_{2,1}\, a^*_{2,2}\, a^*_{2,3} & \ddots & & \\ & \ddots & \ddots & a^*_{n-1,n} \\ 0 & a^*_{n,n-1} & & a^*_{n,n} \end{pmatrix}. \qquad (8.4.2)$$

A^* называется *трехдиагональной матрицей*; для нее $a_{i,j} = 0$, если $|i-j| > 1$.

8.4.2. Приведение матрицы к почти треугольному виду. Приведение матрицы A к виду (8.4.1) будем осуществлять с помощью ортогональных преобразований T, которые, как известно, сохраняют длину векторов, т. е.

$$\| Tx \|_2 = \| x \|_2 \quad \text{или} \quad (Tx, Tx) = (x, x).$$

Напомним, что матрица ортогонального преобразования T обладает свойством

$$T^{-1} = T',$$

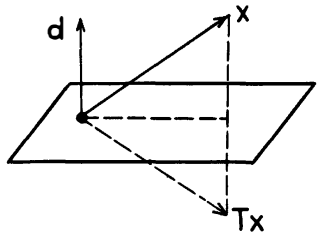

Рис. 8.8

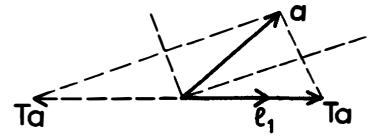

Рис. 8.9

где T' — транспонированная к T матрица; кроме того, произведение ортогональных матриц есть ортогональная матрица.

В качестве матриц T будем рассматривать матрицы отражения. Матрица отражения T, порожденная вектором $d=(d_1, d_2, ..., d_n)$, определяется формулой

$$T=E-2dd'=\begin{pmatrix}1 & & 0\\ & \ddots & \\ 0 & & 1\end{pmatrix}-2\begin{pmatrix}d_1^2 & d_1d_2 & \dots & d_1d_n\\ d_2d_1 & d_2^2 & \dots & d_2d_n\\ \dots\dots & \dots\dots & \dots & \dots\dots\\ d_nd_1 & d_nd_2 & \dots & d_n^2\end{pmatrix}, \quad (8.4.3)$$

где $\|d\|_2=1$.

Геометрически матрица T задает отражение вектора x относительно плоскости с единичным нормальным вектором d (рис. 8.8).

Заметим, что матрица T симметрична, ортогональна: $Td=-d$; если вектор x ортогонален d, то $Tx=x$.

Построим элементарное отражение T, которое переводит заданный вектор $a=(a_1, a_2, ..., a_n)$ в вектор, коллинеарный вектору $l_1=(1, 0, ..., 0)$. Таких отражений два (соответствуют ниже знакам $\pm$) (рис. 8.9):

$$Ta=\pm\|a\|_2e_1.$$

Вектор d, порождающий искомое отражение, задается формулами

$$d=c(a_1\mp\|a\|_2, a_2, ..., a_n), \quad (8.4.4)$$

где константа c определяется из условия

$$\|d\|_2=1. \quad (8.4.5)$$

Объединяя (8.4.4), (8.4.5) и (8.4.3), получим искомое отражение T.

Для однозначного приведения вещественной матрицы A к виду (8.4.1) обычно используют дополнительное требование, состоящее в том, чтобы значения поддиагонали в (8.4.1) были одного знака, например

$$a^*_{i,i-1}\geqslant 0, \quad 2\leqslant i\leqslant n. \quad (8.4.6)$$

Это условие выделяет определенный знак в (8.4.4) и единственное элементарное отражение.

Алгоритм приведения к почти треугольной матрице следующий.

1-й шаг. Определим матрицу n-го порядка:

$$S_1=\begin{pmatrix}1 & 0\\ 0 & T_1\end{pmatrix},$$

где T_1 — матрица $(n-1)$-го порядка — отражение, переводящее вектор

$$\begin{pmatrix} a_{2,1} \\ \vdots \\ a_{n,1} \end{pmatrix}$$

в вектор, коллинеарный $l_1=(1, 0, ..., 0)$. Матрица

$$A^*=S_1AS_1^{-1}=S_1AS_1$$

имеет вид

$$A^*=\begin{pmatrix} * & * & * & ... & * & * \\ * & * & * & ... & * & * \\ 0 & * & * & ... & * & * \\ 0 & * & * & ... & * & * \\ ... & ... & ... & ... & ... & ... \\ 0 & * & * & ... & * & * \end{pmatrix}, \quad a^*_{2,1}\geqslant 0,$$

где знаком $*$ обозначены элементы, которые могут отличаться от нуля.

2-й шаг. Определим матрицу n-го порядка:

$$S_2=\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & T_2 \end{pmatrix},$$

где T_2 — матрица $(n-2)$-го порядка — отражение, переводящее вектор

$$\begin{pmatrix} a^*_{3,2} \\ \vdots \\ a^*_{n,2} \end{pmatrix}$$

в вектор, коллинеарный $l_1=(1, 0, ... 0)$. Матрица

$$A^*=S_2S_1AS_1S_2$$

имеет вид

$$A^*=\begin{pmatrix} * & * & * & ... & * & * \\ * & * & * & ... & * & * \\ 0 & * & * & ... & * & * \\ 0 & 0 & * & ... & * & * \\ ... & ... & ... & ... & ... & ... \\ 0 & 0 & * & ... & * & * \end{pmatrix}, \quad a^*_{3,2}\geqslant 0,$$

Продолжая этот процесс, после $n-2$ шагов получим

$$A^*=S_{n-2} ... S_1AS_1 ... S_{n-2},$$

где A^* — матрица (8.4.1), для которой выполняются неравенства (8.4.6).

Если на каком-либо шаге i получился нулевой вектор, для которого строится отражение, то полагают $S_i=E$. Окончательно ортогональное преобразование T матрицы A к почти треугольному виду определяется произведением ортогональных матриц S_i:

$$T=S_{n-2} ... S_1.$$

Если матрица A симметрична, то промежуточные матрицы и конечная оказываются также симметричными; следовательно, почти треугольная форма таких матриц является трехдиагональной (8.4.2).

8.4.3. Разложение матрицы в произведение ортогональной и треугольной матриц. Будем считать, что матрица A приведена к почти треугольному виду (8.4.1). Найдем разложение матрицы A в произведение:

$$A=QR, \tag{8.4.7}$$

где Q — ортогональная матрица, R — верхняя треугольная матрица:

$$R=\begin{pmatrix} * & * & \dots & * \\ & * & \dots & * \\ 0 & & \ddots & \\ & & & * \end{pmatrix}.$$

Определим матрицу элементарного вращения:

$$P_i=\begin{pmatrix} 1 & & & & & 0 \\ & \ddots & & & & \\ & & 1 & & & \\ & & & \sin\theta_i\cos\theta_i & \dots\dots & \\ & & & -\cos\theta_i\sin\theta_i & & \\ 0 & & & & 1 & \\ & & & & & \ddots \\ & & & & & & 1 \end{pmatrix} \begin{matrix} \\ \\ \\ i \\ \\ \\ \\ \\ \end{matrix} \quad i.$$

Можно проверить, что P_i, $1\leqslant i\leqslant n-1$, — ортогональные матрицы.

Умножая матрицу A вида (8.4.1) на матрицу P_1, получим

$$\begin{pmatrix} \sin\theta_1\cos\theta_1 & & & \\ -\cos\theta_1\sin\theta_1 & 0 & & \\ & 1 & & \\ 0 & & \ddots & \\ & & & 1 \end{pmatrix} \begin{pmatrix} a_{1,1} & \dots & a_{1,n} \\ a_{2,1} & \dots & a_{2,n} \\ & \ddots & \vdots \\ 0 & & \\ & & a_{n,n-1}a_{n,n} \end{pmatrix} =$$

$$\begin{pmatrix} a_{1,1}\sin\theta_1+a_{2,1}\cos\theta_1 & \dots & a_{1,n}\sin\theta_1+a_{2,n}\cos\theta_1 \\ -a_{1,1}\cos\theta_1+a_{2,1}\sin\theta_1 & \dots & -a_{1,n}\cos\theta_1+a_{2,n}\sin\theta_1 \\ a_{3,2} & \dots & a_{3,n} \\ & \ddots & \vdots \\ & & a_{n,n-1}\quad a_{n,n} \end{pmatrix}.$$

Выберем угол θ_1 таким, чтобы поддиагональный элемент в первом столбце обратился в нуль:

$$-a_{1,1}\cos\theta_1+a_{2,1}\sin\theta_1=0,\quad \theta_1=\operatorname{arctg}(a_{1,1}/a_{2,1}).$$

В результате умножения матрицы A на P_1, P_2, ..., P_{n-1} и соответствующего выбора углов вращения θ_i все поддиагональные элементы матрицы A можно обратить в нуль. Таким образом, получим

$$P_{n-1}P_{n-2}\dots P_1A=R, \tag{8.4.8}$$

где R — треугольная матрица. Из (8.4.8) имеем

$$A=QR,$$
$$Q=P'_1P'_2\ \ldots\ P'_{n-1},$$

т. е. (8.4.7). Матрица Q — ортогональная как произведение ортогональных матриц.

8.4.4. *QR*-алгоритм определения собственных значений и векторов. Пусть матрица A приведена к почти треугольному виду (8.4.1). Если бы поддиагональ в (8.4.1) обладала свойством

$$a^*_{i,i-1}a^*_{i+1,i}=0,\ \ 2\leqslant i\leqslant n-1,$$

т. е. в матрице A отсутствовали бы соседние ненулевые поддиагональные элементы, то матрица A имела бы блочно-треугольную форму

$$A^*=\begin{pmatrix} A^*_{1,1} & A^*_{1,2} & \ldots & A^*_{1,n} \\ 0 & A^*_{2,2} & \ldots & A^*_{2,n} \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & A^*_{n,n} \end{pmatrix}. \qquad (8.4.9)$$

Здесь $A^*_{i,i}$ — квадратные блоки порядка 1 или 2. Блоки второго порядка соответствуют ненулевым элементам поддиагонали.

Так как A^* и A имеют одни и те же собственные значения, то блоки $A^*_{i,i}$ первого порядка — это собственные значения A. Собственные значения блоков $A^*_{i,i}$ второго порядка — это также собственные значения A, но среди них могут быть комплексные числа.

Таким образом, приведение матрицы A к форме (8.4.9) полностью решает проблему собственных значений. Собственные векторы после приведения к (8.4.9) также легко определяются.

Однако для матрицы A, вообще говоря, не существует конечного вычислительного процесса над ее элементами, приводящего A к форме (8.4.9). Рассматриваемый здесь QR-алгоритм дает последовательность матриц $A^{(k)}$ почти треугольного вида (8.4.1) таких, что

$$\lim_{k\to\infty} a^{(k)}_{i,i-1}a^{(k)}_{i+1,i}=0.$$

Когда сходящиеся к нулю поддиагональные элементы матриц $A^{(k)}$ будут иметь величину меньше заданной точности, QR-алгоритм завершается.

Каждый шаг QR алгоритма состоит в переходе от матрицы $A^{(k)}$ к матрице $A^{(k+1)}$, $k=0, 1, \ldots, A^{(0)}=A$ по следующему правилу.

1. Матрица $A^{(k)}$ разлагается в произведение ортогональной и треугольной матриц

$$A^{(k)}=Q^{(k)}R^{(k)} \qquad (8.4.10)$$

по схеме (8.4.3).

2. Матрица $A^{(k+1)}$ определяется перестановкой сомножителей в (8.4.10):

$$A^{(k+1)}=R^{(k)}Q^{(k)},\ \ k=0, 1, 2, \ldots \qquad (8.4.11)$$

Из (8.4.10) определим

$$R^{(k)}=\left(Q^{(k)}\right)^{-1}A^{(k)}.$$

Отсюда

$$A^{(k+1)}=\left(Q^{(k)}\right)^{-1}A^{(k)}Q^{(k)},$$

или

$$A^{(k+1)}=\left(Q^{(k)}\right)^{-1}\ldots\left(Q^{(0)}\right)^{-1}AQ^{(0)}\ldots Q^{(k)}.$$

Обозначим $P^{(k)}=Q^{(0)}\ldots Q^{(k)}$. Тогда QR-алгоритм можно записать в форме

$$A^{(k+1)}=\left(P^{(k)}\right)^{-1}AP^{(k)}.$$

Это соотношение указывает, что собственные значения матриц $A^{(k)}$ и A совпадают.

Сформулируем утверждение о сходимости QR алгоритма.

Теорема 8.10. *Пусть матрица A невырождена, имеет различные собственные значения λ_i, $1\leqslant i\leqslant n$, модули $|\lambda_i|$ которых, за исключением комплексно-сопряженных, также различны и упорядочены в порядке невозрастания, тогда при $k\to\infty$*

$$a_{i,i-1}^{(k)}a_{i+1,i}^{(k)}=O\left(\left|\frac{\lambda_{i+1}}{\lambda_{i-1}}\right|^{k}\right),\quad 2\leqslant i\leqslant n-1.$$

8.4.5. Применение библиотечных программ. Рассмотрим следующую задачу: определить собственные значения матрицы

$$A=\begin{pmatrix}1&2&3\\2&4&5\\3&5&6\end{pmatrix}.$$

Матрица A — симметричная, поэтому для решения поставленной задачи применяется программа B0A0:

```
      REAL A(3,3),G(3),W(3)
      INTEGER K,N,I
      DATA A/1.,2.,3.,2.,4.,5.,3.,5.,6./
      DATA K,N,I/3,3,0/
C     ОБРАЩЕНИЕ К ПРОГРАММЕ B0A0
      CALL B0A0(A,K,N,G,W,I)
C     ВЫВОД НА ТЕРМИНАЛ
      WRITE (5,1) G
1     FORMAT (2X,3E13.6)
      END
```

Рассмотрим задачу определения собственных значений и собственных векторов матрицы A:

$$A=\begin{pmatrix}1&2&3\\4&5&6\\7&8&9\end{pmatrix}.$$

Для ее решения применим программу B0A3:

```
      REAL A(3,3),G1(3),G2(3),X1(3,3),X2(3,3)
      INTEGER K,N,L1,L2,J(3),I
      DATA A/1.,4.,7.,2.,5.,8.,3.,6.,9./
      DATA K,N,L1,L2,I/3,3,3,3,0/
C     ОБРАЩЕНИЕ К ПРОГРАММЕ B0A3
      CALL B0A3(A,K,N,G1,G2,X1,L1,X2,L2,J,I)
C     ВЫВОД НА ТЕРМИНАЛ СОБСТВЕННЫХ ЗНАЧЕНИЙ
      WRITE (5,1) (G(I),G2(I),I=1,3)
1     FORMAT (2X,'RE',E13.6,2X,'IM',E13.6)
C     ВЫВОД НА ТЕРМИНАЛ СОБСТВЕННЫХ ВЕКТОРОВ
      WRITE (5,2) ((X(J,I),X2(J,I),J=1,3)I=1,3)
2     FORMAT (2X,E13.6,2X,E13.6)
      END
```

● 8.5. Линейная оптимизация

Рассмотрим задачи, которые в математической литературе обычно называют *линейным программированием*. Термин «линейное программирование» появился до широкого использования ЭВМ и в этой книге не употребляется из-за его несоответствия современному пониманию смысла программирования.

8.5.1. Постановка канонической задачи. Каноническая задача линейной оптимизации формулируется следующим образом: найти вектор $x=(x_1, \ldots, x_n)$, доставляющий

$$\min\Phi(x)=c_1x_1+c_2x_2+\ldots+c_nx_n \qquad (8.5.1)$$

при условиях

$$\begin{aligned} &a_{1,1}x_1+a_{1,2}x_2+\ldots+a_{1,n}x_n=b_1,\\ &a_{2,1}x_1+a_{2,2}x_2+\ldots+a_{2,n}x_n=b_2,\\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ &a_{m,1}x_1+a_{m,2}x_2+\ldots+a_{m,n}x_n=b_m, \end{aligned} \qquad (8.5.2)$$

$$x_i\geqslant 0,\ 1\leqslant i\leqslant n,\ b_j\geqslant 0,\ 1\leqslant j\leqslant m. \qquad (8.5.3)$$

Задача (8.5.1)—(8.5.3) сокращенно записывается в матрично-векторной форме так:

$$\begin{aligned} &\min(c, x),\\ &Ax=b,\\ &x\geqslant 0,\ b\geqslant 0, \end{aligned} \qquad (8.5.4)$$

где вещественные матрица $A=(a_{i,j})$, векторы c_i, b_j, $1\leqslant i\leqslant n$, $1\leqslant j\leqslant m$, заданы; строки матрицы A линейно независимы, откуда, в частности, следует $m\leqslant n$. Неотрицательность элементов вектора b легко получить, умножив, если это необходимо, соответствующие неравенства в (8.5.2) на -1.

В общей постановке задача линейной оптимизации может не иметь каноническую форму. Однако, как будет показано ниже, введением дополнительных переменных в вектор x каждая задача сводится к канонической.

Функция $\Phi(x)$ называется *целевой функцией задачи*, система (8.5.2), (8.5.3) — *системой ограничений.*

В общей постановке задачи линейной оптимизации, возможно, ищется максимум целевой функции $\max \Phi(x)$. Тогда замена целевой функции на $-\Phi(x)$ сводит исходную задачу к минимизации $\min(-\Phi(x))$. Кроме того, в соотношениях типа (8.5.2) могут присутствовать неравенства, например условие

$$a_1 x_1 + a_2 x_2 + \ldots + a_n x_n \leqslant b$$

или

$$a_1 x_1 + a_2 x_2 + \ldots + a_n x_n \geqslant b.$$

Тогда каждое неравенство можно заменить двумя соотношениями вида

$$a_1 x_1 + a_2 x_2 + \ldots + a_n x_n + x_{n+1} = b,$$
$$x_{n+1} \geqslant 0,$$

или

$$a_1 x_1 + a_2 x_2 + \ldots + a_n x_n - x_{n+2} = b,$$
$$x_{n+2} \geqslant 0;$$

целевую функцию — записать в форме

$$\Phi(x) = c_1 x_1 + \ldots + c_n x_n + 0 x_{n+1} + 0 x_{n+2}$$

и таким образом привести общую задачу линейной оптимизации к каноническому виду.

8.5.2. Примеры задач линейной оптимизации. Следующая задача об определении максимальной прибыли производства является типичной задачей линейной оптимизации. Пусть для изготовления n видов продукции в количестве x_1, x_2, ..., x_n расходуется m видов сырья. Расход сырья i-го вида на единицу j-го вида продукции равен $a_{i,j}$, запас сырья i-го вида ограничен величиной b_i. Каждая единица продукции i-го вида дает прибыль c_i руб. Задача поиска вектора $x = (x_1, \ldots, x_n)$ — количества продукции каждого вида, дающего максимальную прибыль, — принимает следующую форму: найти x, доставляющий

$$\max \Phi(x) = c_1 x_1 + c_2 x_2 + \ldots + c_n x_n$$

при ограничениях

$$\sum_{j=1}^{n} a_{i,j} x_j \leqslant b_i, \quad 1 \leqslant i \leqslant m,$$
$$x_i \geqslant 0, \quad 1 \leqslant i \leqslant n.$$

Таким образом получили задачу линейной оптимизации, которая введением дополнительных переменных легко приводится к канонической форме.

Второй пример линейной оптимизации дает так называемая транспортная задача. Пусть в i-м пункте, $1 \leqslant i \leqslant m$, отправления

имеется d_i количество груза, в j-м пункте, $1 \leqslant j \leqslant n$, назначения требуется b_j количество этого груза. Пусть $c_{i,j}$ — стоимость перевозки единицы груза из i-го в j-й пункт. Предположим, что

$$\sum_{i=1}^{m} d_i = \sum_{j=1}^{n} b_j,$$

т. е. весь груз должен быть вывезен на пункты назначения.

Транспортная задача состоит в поиске $x = x_{i,j}$, $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n$, доставляющего

$$\min \Phi(x) = \sum_{i,j=1}^{m,n} c_{i,j} x_{i,j}$$

(минимизация транспортных расходов) при естественных условиях

$$\sum_{j=1}^{n} x_{i,j} = d_i,$$

$$\sum_{i=1}^{m} x_{i,j} = b_j,$$

$$x_{i,j} \geqslant 0.$$

Размерность неизвестного вектора x в транспортной задаче равна mn. Нетрудно заметить, что получается каноническая форма задачи линейной оптимизации.

8.5.3. Геометрическая интерпретация. Рассмотрим каноническую задачу линейной оптимизации на плоскости: найти

$$\min \Phi(x) = (c_1 x_1 + c_2 x_2)$$

при ограничениях

$$a_{1,1} x_1 + a_{1,2} x_2 = b_1,$$
$$a_{2,1} x_1 + a_{2,2} x_2 = b_2,$$
$$x_1 \geqslant 0, \quad x_2 \geqslant 0.$$

Заметим, что линейная функция $\Phi(x) = c_1 x_1 + c_2 x_2$ на плоскости задает семейство прямых (рис. 8.10) уравнением

$$\Phi(x) = p.$$

Неравенства $x_1 \geqslant 0$, $x_2 \geqslant 0$ выделяют I квадрант плоскости (x_1, x_2) (рис. 8.10).

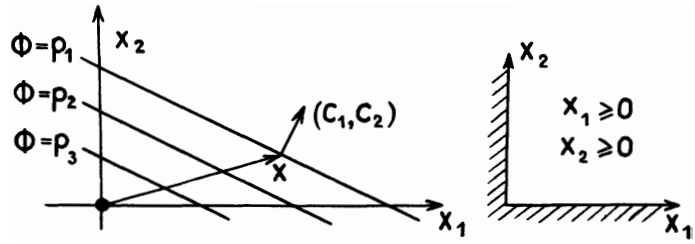

Рис. 8.10

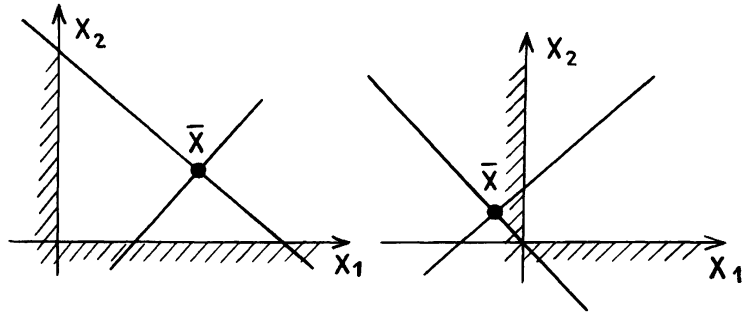

Рис. 8.11

Ограничения в форме равенств могут быть следующего вида:

1) отсутствуют;
2) одно ограничение;
3) два ограничения.

Предположение о линейной независимости строк матрицы A эквивалентно в случае двух ограничений существованию единственного решения системы уравнений

$$Ax=b.$$

Обозначим решение $\bar{x}=(\bar{x}_1,\ \bar{x}_2)$. Если выполняются неравенства

$$\bar{x}_1 \geqslant 0, \quad \bar{x}_2 \geqslant 0,$$

то это и есть решение задачи линейной оптимизации, а оптимальное значение равно

$$\Phi(\bar{x})=c_1\bar{x}_1+c_2\bar{x}_2.$$

Если выполняются неравенства

$$\bar{x}_1<0 \quad \text{или} \quad \bar{x}_2<0,$$

то задача неразрешима (рис. 8.11). Заметим, что два линейно независимых ограничения типа равенства (в многомерном случае $m=n$) приводят к тому, что исходная задача оказывается неоптимизационной. Ограничения однозначно определяют единственное решение $\bar{x}$, а целевая функция $\Phi(x)$ принимает «навязанное» ей значение $\Phi(\bar{x})$.

В случае одного ограничения типа равенства, описывающего множество точек на прямой

$$a_1x_1+a_2x_2=b,$$

возможны следующие варианты пересечения этой прямой с осями координат I квадранта (рис. 8.12):

1) прямая отсекает два отрезка;
2) прямая отсекает один отрезок;
3) прямая не пересекается с осями.

Если прямая не пересекается с осями координат I квадранта, то задача неразрешима.

Если прямая отсекает один отрезок, то возможны три варианта. Проекция вектора градиента целевой функции $\operatorname{grad}\Phi(x)=(c_1,\ c_2)$:

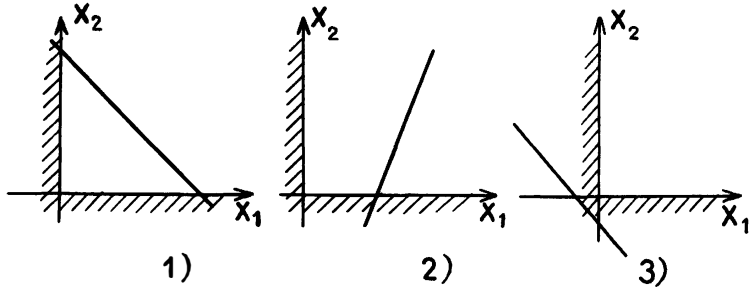

Рис. 8.12

1) направлена внутрь квадранта;
2) направлена вне квадранта;
3) равна нулю.

Так как вектор градиента направлен в сторону возрастания $\Phi(x)$, то нетрудно понять, что в случае 2) целевая функция $\Phi(x)$ не ограничена снизу; вдоль прямой $a_1x_1+a_2x_2=b$ значения $\Phi(x)$ убывают и $\Phi(x)\to-\infty$ (рис. 8.13). В случае 3) вектор $(c_1,\ c_2)$ коллинеарен вектору $(a_1,\ a_2)$. Следовательно, для того чтобы выполнялось ограничение, необходимо совпадение прямых $\Phi(x)=p$ и $a_1x_1+a_2x_2=b$. Отсюда получаем: оптимальными являются все точки пересечения прямой ограничения с прямой $\Phi(x)=p$, лежащие внутри I квадранта (рис. 8.13). В случае 1), поскольку вектор $\operatorname{grad}\Phi(x)$ направлен в сторону возрастания $\Phi(x)$, целевая функция $\Phi(x)$ достигает минимального значения, когда прямая $\Phi(x)=p$ проходит через точку $\bar{x}$ пересечения прямой ограничений с осью координат I квадранта (рис. 8.13). Следовательно, оптимальной является единственная точка $\bar{x}$.

Если прямая $a_1x_1+a_2x_2=b$ отсекает два отрезка на осях координат I квадранта, то в зависимости от направления проекции градиента на эту прямую возможны (рис. 8.14) три варианта: 1) Проекция направлена в сторону точки пересечения с осью Ox_2; тогда оптимальная точка — точка пересечения с осью Ox_1. 2) Проекция направлена в сторону точки пересечения с осью Ox_1; тогда оптимальная точка — это точка пересечения с осью Ox_2. 3) Проекция равна нулю; следовательно, прямая $\Phi(x)=p$ должна совпадать

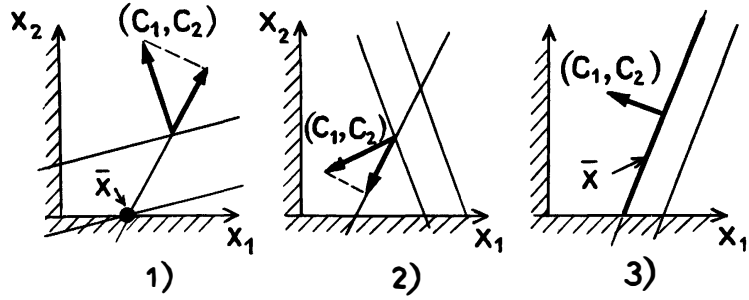

Рис. 8.13

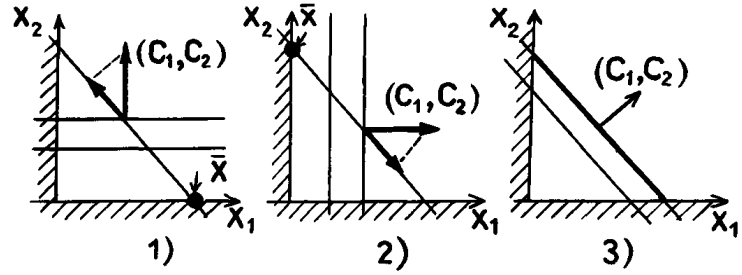

Рис. 8.14

с прямой ограничения, отсюда оптимальными являются все точки прямой ограничений, лежащие внутри I квадранта (рис. 8.14).

Наконец, если ограничения в форме равенств отсутствуют, то возможны (рис. 8.15) следующие оптимальные решения в зависимости от направления вектора $\operatorname{grad}\Phi(x)$:

1) единственная оптимальная точка $\bar{x}_1=0$, $\bar{x}_2=0$;

2), 3) бесконечное множество оптимальных точек (оси координат);

4) целевая функция в I квадранте не ограничена снизу.

Результатом геометрических рассмотрений задачи линейной оптимизации на плоскости являются следующие выводы, которые справедливы и в многомерной задаче.

Линейная целевая функция $\Phi(x)$ может достигать единственного минимума в точке $\bar{x}$ только на границе области $x_i \geqslant 0$ (рис. 8.13, *1*); функция не достигает минимума (рис. 8.13, *2*; 8.15, *4*), а значит, задача линейной оптимизации неразрешима. Наконец, возможна ситуация наличия бесконечного множества решений (рис. 8.13, *3*; 8.14, *3*; 8.15, *2*, *3*).

Мы не будем рассматривать неразрешимые задачи, поскольку в этом случае следует обратиться к технической постановке проблемы, а также задачи с бесконечным множеством решений, так как они неустойчивы. Возмущение коэффициентов c_i целевой функции $\Phi(x)$ приводит такую задачу либо к единственной точке минимума, либо к неразрешимой задаче.

8.5.4. Симплекс-метод. Допустимым решением задачи (8.5.1)—(8.5.3) назовем вектор $x=(x_1, \ldots, x_n)$, удовлетворяющий ограничениям (8.5.2), (8.5.3). Допустимое решение, минимизирующее $\Phi(x)$, является оптимальным. Напомним, что ранг матрицы A равен m.

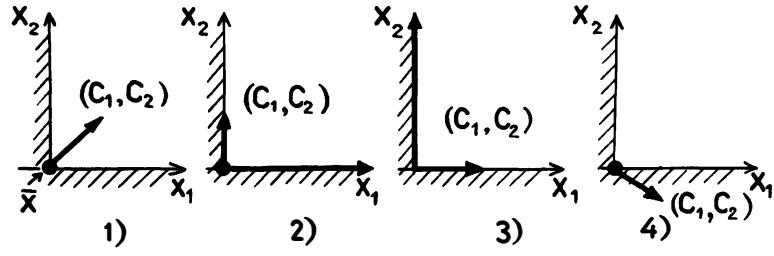

Рис. 8.15

Предположим, что решение задачи существует и единственно. Тогда оптимальное решение может быть получено из каких-то $n-m$ уравнений (8.5.2) и каких-то $n-m$ равенств

$$x_k=0, \quad 1\leqslant k\leqslant n.$$

Но каких именно, неизвестно. Симплекс-метод представляет собой алгоритм решения этого вопроса.

1-й шаг: выбор базисного решения. Выберем $n-m$ свободных неизвестных в (8.5.2), а остальные m переменных (базисные) выразим через них. Пусть отличный от нуля минор m-го порядка расположен в левом верхнем углу матрицы A. Тогда из (8.5.2) можно найти

$$\begin{aligned}
&x_1+\alpha_{1,m+1}x_{m+1}+\ldots+\alpha_{1,n}x_n=\beta_1,\\
&x_2+\alpha_{2,m+1}x_{m+1}+\ldots+\alpha_{2,n}x_n=\beta_2,\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&x_m+\alpha_{m,m+1}x_{m+1}+\ldots+\alpha_{m,n}x_n=\beta_m.
\end{aligned} \tag{8.5.5}$$

Предположим, что

$$\beta_i\geqslant 0, \quad 1\leqslant i\leqslant m. \tag{8.5.6}$$

Подставляя (8.5.5) в (8.5.1), получим

$$\Phi(x_{m+1},\ldots,x_n)=\gamma_{m+1}x_{m+1}+\ldots+\gamma_n x_n+\Phi_0, \tag{8.5.7}$$

где константы γ_i, Φ_0 вычисляются по c_i, $\alpha_{i,j}$, β_i. Условие (8.5.6) позволяет принять вектор x с компонентами

$$x_i=\begin{cases}\beta_i, & 1\leqslant i\leqslant m,\\ 0, & m+1\leqslant i\leqslant n,\end{cases} \tag{8.5.8}$$

в качестве допустимого базисного решения.

Выбор таких базисных переменных, чтобы выполнялось условие (8.5.6), нетривиален. Соответствующая процедура описана ниже. Если в (8.5.7) все $\gamma_i>0$, $m+1\leqslant i\leqslant n$, то (8.5.8) — единственное оптимальное решение, оптимальное значение $\Phi=\Phi_0$.

Пусть среди коэффициентов γ_i есть отрицательные числа, γ_J минимальное из них. Тогда выполняется второй шаг.

2-й шаг: замена допустимого базисного решения (8.5.8) новым с уменьшением целевой функции.

Рассмотрим числа

$$\frac{\beta_i}{\alpha_{i,J}}, \quad 1\leqslant i\leqslant m, \quad m+1\leqslant J\leqslant n.$$

Выберем из них положительные, а среди положительных — наибольшее. Пусть это число

$$x_J=\frac{\beta_I}{\alpha_{I,J}}. \tag{8.5.9}$$

Новое базисное решение определяется формулой (8.5.9) (x_J вводится в базисные переменные) следующими соотношениями:

$$x_i=\begin{cases}\beta_i-\alpha_{I,J}x_J\geqslant 0, & 1\leqslant i\leqslant m,\\ 0 \quad m+1\leqslant i\leqslant n, & i\neq J.\end{cases}$$

Заметим, что $x_I=0$ (x_I выводится из базисных переменных). Целевая функция на этом решении равна

$$\Phi=\Phi_0+\gamma_J\frac{\beta_I}{\alpha_{I,J}}<\Phi_0$$

и в силу выбора J, I ее значение на новом базисном решении меньше, чем на старом.

Вводя в (8.5.5) строку

$$x_J=\frac{\beta_I}{\alpha_{I,J}}-\frac{1}{\alpha_{I,J}}\left(x_I+\sum_{\substack{j=m+1\\ j\neq J}}^{n}\alpha_{I,j}x_j\right),$$

получаем каноническую форму задачи относительно новых базисных переменных.

Повторяем второй шаг до тех пор, пока все коэффициенты при свободных неизвестных в целевой функции не примут положительные значения.

8.5.5. Первый шаг симплекс-метода. Симплекс-метод, как указывалось выше, предполагает на первом шаге выбор базисных переменных x_1, ..., x_m таким образом, чтобы выполнялись условия (8.5.6). Для такого выбора можно вводить дополнительные переменные. Этот подход позволяет установить существование допустимых решений. Однако для его реализации требуется решать еще одну задачу линейной оптимизации тем же симплекс-методом.

Рассмотрим исходную задачу (8.5.1)—(8.5.3). Для нее построим вспомогательную задачу: найти

$$\min\psi(x_{n+1},\ ...,\ x_{n+m})=x_{n+1}+x_{n+2}+...+x_{n+m}$$

для неизвестного вектора x с m дополнительными переменными

$$x=(x_1,\ ...,\ x_n,\ x_{n+1},\ ...,\ x_{n+m})$$

и условиями

$$\begin{gathered}a_{1,1}x_1+a_{1,2}x_2+...+a_{1,n}x_n+x_{n+1}=b_1,\\ a_{2,1}x_1+a_{2,2}x_2+...+a_{2,n}x_n+x_{n+2}=b_2,\\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ a_{m,1}x_1+a_{m,2}x_2+...+a_{m,n}x_n+x_{n+m}=b_m,\\ x_i\geqslant 0,\quad 1\leqslant i\leqslant n+m.\end{gathered}$$

Принимая x_1, x_2, ..., x_n за свободные переменные, а x_{n+1}, x_{n+2}, ..., x_{n+m} — за базисные, получаем базисное допустимое решение вспомогательной задачи:

$$\begin{aligned}&x_{n+s}=b_s\geqslant 0,\quad 1\leqslant s\leqslant m,\\ &x_i=0,\quad 1\leqslant i\leqslant n.\end{aligned}$$

Затем, применяя последовательно второй шаг симплекс-метода, находим оптимальное решение вспомогательной задачи. Если это решение таково, что оптимальное значение $\psi=0$, то оно определяет

допустимое решение исходной задачи; если $\psi > 0$, то исходная задача не имеет допустимых решений.

8.5.6. Применение программы B1A0. Решим задачу линейной оптимизации

$$\max \Phi(x) = 10x_1 - x_2 - 9x_3 - 8x_4$$

при ограничениях

$$\begin{aligned} -2x_1 + x_2 + 3x_3 + x_4 &= 2, \\ -5x_1 + 2x_2 + 3x_4 &= 5, \\ 7x_1 - 4x_2 + x_3 + 4x_4 &\leqslant 1, \\ 3x_1 + 2x_2 + 5x_3 + 6x_4 &\leqslant 10, \\ x_i \geqslant 0, \quad 1 \leqslant i &\leqslant 4. \end{aligned}$$

Предварительно приведем задачу к каноническому виду, дополнив вспомогательными переменными x_5, x_6. Получим

$$\min \Phi_1(x) = -10x_1 + x_2 + 9x_3 + 8x_4 + 0 \cdot x_5 + 0 \cdot x_6$$

при ограничениях

$$\begin{aligned} -2x_1 + x_2 + 3x_3 + x_4 + 0x_5 + 0x_6 &= 2, \\ -5x_1 + 2x_2 + 0x_3 + 3x_4 + 0x_5 + 0x_6 &= 5, \\ 7x_1 - 4x_2 + x_3 + 4x_4 + x_5 + 0x_6 &= +1, \\ 3x_1 + 2x_2 + 5x_3 + 6x_4 + 0x_5 + x_6 &= 10, \\ x_i \geqslant 0, \quad 1 \leqslant i &\leqslant 6. \end{aligned}$$

Зададим абсолютную точность выполнения неравенств $x_i \geqslant 0$, равную 10^{-5}. Программа может иметь следующий вид:

```
      REAL W(24),U(36),X(6),E,V(6),A(4,6)
      REAL B(4),C(6)
      INTEGER M,N,K(4),I
      DATA A/-2.,-5.,7.,3.,1.,2.,-4.,2.,3.,0.,1.,5.,
     *1.,3.,4.,6.,0.,0.,1.,0.,0.,0.,0.,1./
      DATA B/2.,5.,1.,10./,E/1.E-5/
      DATA C/-10.,1.,9.,8.,0.,0./
      DATA M,N,I/6,6,0/
C     ОБРАЩЕНИЕ К ПРОГРАММЕ B1A0
      CALL B1A0(W,M,U,N,X,K,I,E,V,A,B,C)
C     ВЫВОД НА ТЕРМИНАЛ ЗНАЧЕНИЙ ОПТИМАЛЬНЫХ
C     КООРДИНАТ,
C     НОМЕРОВ КООРДИНАТ, ОПТИМАЛЬНОГО ЗНАЧЕ-
C     НИЯ ЦЕЛЕВОЙ ФУНКЦИИ
      WRITE (5,1) (X(L),L=1,4)
1     FORMAT (2X,6E12.5)
      WRITE (5,2) K
2     FORMAT (2X,4(I2,2X))
      WRITE (5,3) X(5)
3     FORMAT (2X,'FI1=',E12.5)
      END
```

Глава 9

НЕЛИНЕЙНЫЕ УРАВНЕНИЯ. НЕЛИНЕЙНАЯ ОПТИМИЗАЦИЯ

● 9.1. Введение

9.1.1. Сравнение линейных и нелинейных уравнений. Важнейшим свойством линейных функций $y=f(x)$ является следующее: $f(x)$ удовлетворяют принципу суперпозиции. Принцип суперпозиции заключается в том, что если аргумент x есть линейная суперпозиция (комбинация) двух точек

$$x=c_1x_1+c_2x_2,$$

c_1, c_2 — константы, то значение y есть линейная суперпозиция значения $f(x_1)$ и $f(x_2)$, а именно

$$y=f(x)=f(c_1x_1+c_2x_2)=c_1f(x_1)+c_2f(x_2). \tag{9.1.1}$$

Например, пусть x, y — векторы, $f(x)=Ax$, где A — матрица размером $n\times n$. Тогда свойство (9.1.1) — следствие свойств матричных операций

$$A(c_1x_1+c_2x_2)=c_1Ax_1+c_2Ax_2.$$

Нелинейные функции $y=f(x)$ не удовлетворяют принципу суперпозиции, например для $y=x^2$ (9.1.1) не выполняется:

$$(c_1x_1+c_2x_2)^2\neq c_1x_1^2+c_2x_2^2.$$

Этот факт лежит в основе различия линейных и нелинейных уравнений.

Наиболее общий вид нелинейных уравнений, изучаемых в настоящей главе, следующий:

$$\begin{aligned} f_1(x_1, x_2, ..., x_n)&=y_1,\\ f_2(x_1, x_2, ..., x_n)&=y_2,\\ &\dots\dots\dots\dots\\ f_n(x_1, x_2, ..., x_n)&=y_n, \end{aligned} \tag{9.1.2}$$

или в векторной записи

$$f(x)=y, \tag{9.1.3}$$

здесь f — функция (или отображение) с областью определения $D\in E^n$ и областью значений $Q\in E^n$, вектор y задается.

Частным случаем (9.1.3) являются линейные уравнения

$$Ax=y. \tag{9.1.4}$$

Первое отличие (9.1.3) от (9.1.4) состоит в том, что область определения D в (9.1.3) может не совпадать с E^n. Например, функция

$$\ln x = y \tag{9.1.5}$$

имеет область определения $D = \{0 < x < \infty\} \in E^1 = \{-\infty < x < \infty\}$. Поэтому решение уравнения (9.1.5) следует искать только в области D.

Второе отличие (9.1.3) от (9.1.4) состоит в том, что в области определения D при заданном y может существовать несколько решений. Такой ситуации не было в линейных уравнениях.

В линейных уравнениях возможны следующие варианты. Имеется:

1) единственное решение;
2) бесконечное множество решений;
3) решений нет.

Линейные уравнения являются частным случаем нелинейных, поэтому для нелинейных уравнений возможны упомянутые выше три варианта, а также:

4) имеется n решений, $2 \leqslant n < \infty$.

Примером четвертого варианта могут служить задачи определения корней полиномов

$$P_n(x) = 0. \tag{9.1.6}$$

Если полином $P_n(x)$ можно представить в виде

$$P_n(x) = (x - x_1)(x - x_2)\ldots(x - x_n),$$

где x_i — вещественные числа, то уравнение (9.1.6) имеет n решений в E^1.

9.1.2. Геометрическая интерпретация решения системы уравнений. Каждое уравнение в системе

$$\begin{aligned} &f_1(x_1, x_2, \ldots, x_n) = 0, \\ &f_2(x_1, x_2, \ldots, x_n) = 0, \\ &\ldots\ldots\ldots\ldots\ldots\ldots \\ &f_n(x_1, x_2, \ldots, x_n) = 0 \end{aligned} \tag{9.1.7}$$

определяет некоторую (не обязательно непрерывную) поверхность в E^n. Следовательно, решениями (9.1.7) являются точки пересечения этих поверхностей.

Для примера рассмотрим систему двух уравнений с двумя неизвестными

$$f_1 = x_1^2 - x_2 + \alpha = 0,$$

$$f_2 = -x_1 + x_2^2 + \alpha = 0,$$

где α — вещественный параметр, $+1 \leqslant \alpha \leqslant -1$. Если изменять α в указанном интервале, то имеют место следующие случаи (рис. 9.1):

$\alpha = 1$ — решений нет,

$\alpha = 1/4$ — решение единственно,

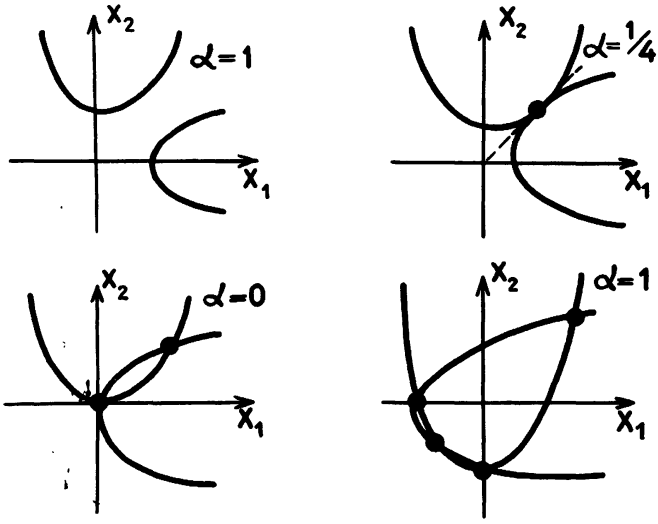

Рис. 9.1

$\alpha=0$ — два решения,

$\alpha=-1$ — четыре решения.

Система уравнений

$$f_1=x_1\sin x_1-x_2=0,$$
$$f_2=x_2^2-x_1+1=0$$

имеет в E^2 бесконечное множество решений (рис. 9.2).

Корень x_* скалярного уравнения $f(x)=0$ интерпретируется как точка пересечения кривой $y=f(x)$ с осью абсцисс (рис. 9.3).

Если нелинейное уравнение записывается в виде

$$x=\varphi(x),$$

то корень x_* такого уравнения представляет собой точку пересечения прямой

$$y=x$$

и кривой

$$y=\varphi(x)$$

(рис. 9.4).

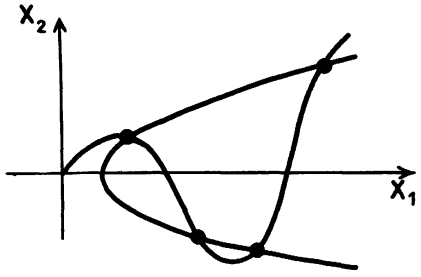

Рис. 9.2

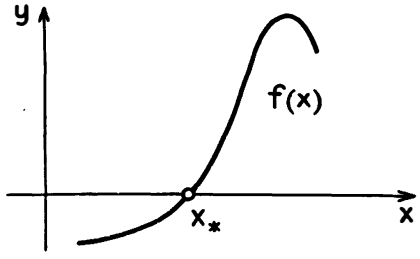

Рис. 9.3

9.1.3. Существование решения и линеаризация. Вопрос о существовании решения системы уравнений (9.1.2) в окрестности точки (x^0, y^0) решает теорема о неявной функции, известная в математическом анализе.

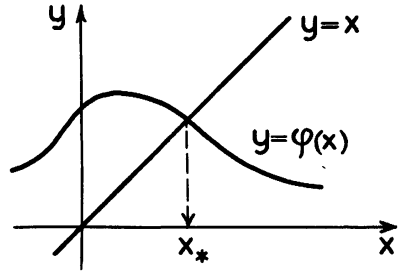

Рис. 9.4

Теорема 9.1. *Пусть функции* $f_i(x_1, \ldots, x_n)$, $1 \leqslant i \leqslant n$, *непрерывно дифференцируемы в окрестности точки* $x^0=(x_1^0, \ldots, x_n^0)$

$$f_i(x_1^0, \ldots, x_i^0)=y_i^0, \quad 1 \leqslant i \leqslant n.$$

Пусть матрица

$$A(x^0)=\begin{pmatrix} \frac{\partial f_1}{\partial x_1}(x_1^0, \ldots, x_n^0) \ldots \frac{\partial f_1}{\partial x_n}(x_1^0, \ldots, x_n^0) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \frac{\partial f_n}{\partial x_1}(x_1^0, \ldots, x_n^0) \ldots \frac{\partial f_n}{\partial x_n}(x_1^0, \ldots, x_n^0) \end{pmatrix}$$

невырождена. Тогда существуют непрерывные функции

$$x_i=z_i(y_1, \ldots, y_n), \quad 1 \leqslant i \leqslant n,$$

в окрестности точки $y^0=(y_1^0, \ldots, y_n^0)$, *обладающие свойствами*

$$f_i(z_1(y_1, \ldots, y_n), \ldots, z_n(y_1, \ldots, y_n))=y_i$$

$$z_i(y_1^0, \ldots, y_n^0)=x_i^0, \quad 1 \leqslant i \leqslant n.$$

Теорема 9.1 дает подход к решению нелинейных систем уравнений, основанный на линеаризации системы (9.1.2), т. е. замене (9.1.2) на «близкую» линейную систему.

Будем предполагать, что фукции $f_i(x_1, \ldots, x_n)$ дважды непрерывно дифференцируемы в E^n. Представим левую часть (9.1.2) отрезком ряд Тейлора с центром в какой-либо точке x^0:

$$f_i(x^0)+\sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}(x^0)(x_j-x_j^0)+O(\|x-x^0\|^2)=y_i^0.$$

Если отбросить в этой формуле остаточный член, то получим линеаризованную в окрестности x^0, y^0 систему уравнений (9.1.2), которую запишем в матричной форме:

$$f(x^0)+A(x^0)(x-x^0)=y^0. \tag{9.1.8}$$

Поскольку точное решение исходной системы уравнений неизвестно, в общем случае

$$f(x^0) \neq y^0.$$

Пусть

$$\det A(x^0) \neq 0.$$

Тогда численно решаем систему линейных уравнений (9.1.8), получаем решение линеаризованной системы

$$x=A^{-1}(x^0)(y^0-f(x^0))+x^0. \tag{9.1.9}$$

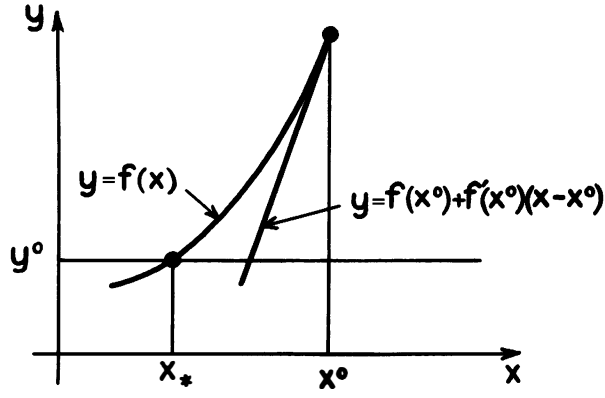

Рис. 9.5

Погрешность численного решения (9.1.8) и погрешность, связанная с отбрасыванием остаточного члена, дают полную погрешность (9.1.9), если принять решение линеаризованного уравнения за приближенное решение (9.1.2).

Очевидно, что погрешность линеаризации определяется числом обусловленности матрицы $A(x^0)$ и выбором нулевого приближения x^0.

Грубое приближение к x^0 обычно получают из анализа научно-технической задачи.

Линеаризация лежит в основе итерационного метода Ньютона (см. **9.3**).

В скалярном случае линеаризация эквивалентна замене нелинейной функции $f(x)$ на линейную, проходящую через точку $f(x^0)$ и касательную к $f(x)$ в точке x^0 (рис. 9.5). Невырожденность $A(x^0)$ эквивалентна условию $f'(x^0)\neq 0$.

● 9.2. Сжимающие отображения

9.2.1. Определение сжимающего отображения. Рассмотрим линейное n-мерное вещественное пространство E^n; определим в E^n какую-либо норму $\|x\|$ элемента $x=(x_1, ..., x_n)$.

Пусть задано, вообще говоря, нелинейное отображение $y=\varphi(x)$, например, с помощью n функций

$$\begin{aligned} y_1&=\varphi_1(x_1, ..., x_n),\\ y_2&=\varphi_2(x_1, ..., x_n),\\ &\cdots\cdots\cdots\cdots\\ y_n&=\varphi_n(x_1, ..., x_n), \end{aligned}$$

определенное на всем пространстве E^n.

Отображение $\varphi(x)$ переводит E^n в себя, если для любого элемента x, принадлежащего E^n, элемент $y=\varphi(x)$ также принадлежит E^n. Например, отображение $y=\sin x$ переводит E^1 в себя, отображение $y=\sqrt{\sin x}$ этим свойством не обладает.

Отображение $\varphi(x)$, переводящее E^n в себя, называется *сжимающим* в E^n, если для произвольных двух элементов $\bar{x}\bar{\bar{x}} \in E^n$ имеет место неравенство

$$\|\varphi(\bar{x})-\varphi(\bar{\bar{x}})\| \leqslant \alpha \|\bar{x}-\bar{\bar{x}}\|, \qquad (9.2.1)$$

где коэффициент сжатия α удовлетворяет неравенству

$$0<\alpha<1. \qquad (9.2.2)$$

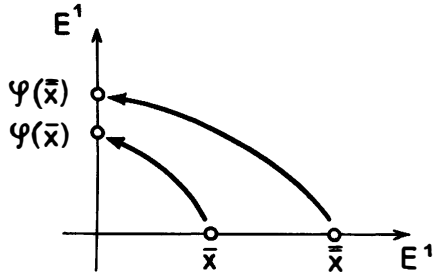

Рис. 9.6

Например, $\varphi(x)=0{,}1\sin x$ — сжимающее отображение в E^1 с нормой $\|x\|=|x|$. Покажем, что $\varphi(x)$ удовлетворяет (9.2.1), (9.2.2). Действительно,

$$|\varphi(\bar{x})-\varphi(\bar{\bar{x}})| = |0{,}1\sin\bar{x}-0{,}1\sin\bar{\bar{x}}| = 0{,}1\,|\sin\bar{x}-\sin\bar{\bar{x}}|.$$

Но из теоремы о среднем имеем

$$|\sin\bar{x}-\sin\bar{\bar{x}}| = |\cos\xi|\,|\bar{x}-\bar{\bar{x}}|, \quad \bar{x}<\xi<\bar{\bar{x}}.$$

Из двух последних соотношений получаем

$$|\varphi(\bar{x})-\varphi(\bar{\bar{x}})| \leqslant 0{,}1\,|\bar{x}-\bar{\bar{x}}|,$$

т. е. неравенство (9.2.1) с коэффициентом сжатия $\alpha=0{,}1$.

Действие сжимающего отображения иллюстрирует рис. 9.6.

Для нелинейных функций $\varphi(x)$ требование, чтобы $\varphi(x)$ было определено на всем пространстве E^n, оказывается слишком ограничительным. Поэтому отображение $\varphi(x)$ часто рассматривают локально, только в шаре S, принадлежащем области определения D отображения. Шар S — это совокупность элементов $x \in E^n$ таких, что $\|x-x^0\| \leqslant r$:

$$S=\{x:\ \|x-x^0\| \leqslant r\}.$$

Здесь x^0 — элемент, E^n — центр шара, r — положительное число — радиус шара (рис. 9.7).

Отображение $\varphi(x)$ переводит шар S в себя, если для любого $x \in S$, элемент $y=\varphi(x)$ также принадлежит шару S. Например, отображение $y=\sqrt{\sin x}$ переводит шар $|x-\pi/2| \leqslant \pi/2$ с центром в точке $x^0=\pi/2$ и радиусом $\pi/2$ в себя.

Отображение $\varphi(x)$, переводящее шар S в E^n, называется *сжимающим* в S, если для произвольных двух элементов $\bar{x}$, $\bar{\bar{x}} \in S$ имеют место неравенства (9.2.1), (9.2.2).

9.2.2. Метод простой итерации. Сжимающие отображения представляют собой замечательный класс отображений, определяющих нелинейные уравнения вида

$$x=\varphi(x). \qquad (9.2.3)$$

Для этих уравнений легко ответить на вопрос о существовании и единственности решения (9.2.3), а также построить последовательность приближений $\{x^{(k)}\}$, сходящихся к решению.

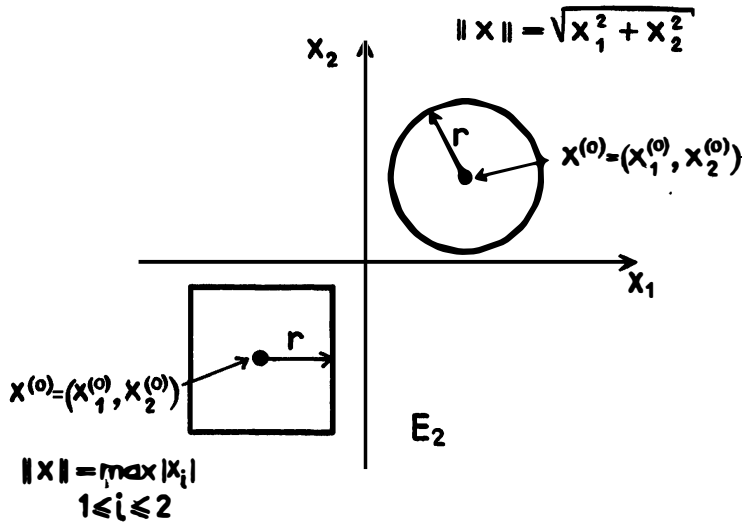

Рис. 9.7

Если уравнение имеет вид

$$f(x)=0, \tag{9.2.4}$$

то оно предварительно должно быть преобразовано к форме (9.2.3), причем таким образом, чтобы $\varphi(x)$ оказалось сжимающим отображением. Общего приема для перехода от (9.2.4) к (9.2.3) не существует, и здесь важен предварительный анализ задачи, например исследование линеаризованного уравнения.

Построим последовательность $\{x^{(k)}\}$ по формуле

$$x^{(k+1)}=\varphi(x^{(k)}), \quad k=0, 1, 2, \ldots, \tag{9.2.5}$$

где x^0 — начальный элемент, который следует задать. Если последовательность $\{x^{(k)}\}$ сходится к x, т. е.

$$\|x^{(k)}-x\| \to 0 \text{ при } k \to \infty,$$

а $\varphi(x)$ — непрерывное отображение, то, переходя в (9.2.5) к пределу при $k \to \infty$, получим, что предел $x=\lim\limits_{k\to\infty} x^{(k)}$ — точное решение уравнения (9.2.3). Таким образом, решение нелинейного уравнения (9.2.3) может быть получено как предел последовательности итераций $\{x^{(k)}\}$.

Приближенное решение (9.2.3) с помощью (9.2.5) называется *методом простой итерации.*

Сравним (9.2.5) с методом простой интерации в системах линейных уравнений (см. **8.3**). Метод простой итерации (9.2.5) является соответствующим методом решения линейных уравнений, если обозначить

$$\varphi(x)=Bx+d,$$

где B — матрица, x, d — векторы соответствующих размерностей. Аналогом достаточного условия сходимости простых итераций в линейном случае ($\|B\| < 1$) оказывается условие сжимаемости ($\alpha < 1$) в нелинейных уравнениях. Этот факт устанавливает следующая теорема.

Теорема 9.2. *Пусть $\varphi(x)$ — сжимающее отображение E^n в себя. Тогда существует единственное решение уравнения* (9.2.3), *к которому сходится последовательность $\{x^{(k)}\}$, получаемая методом простой итерации* (9.2.5) *с любым начальным элементом $x^{(0)}$.*

Доказательство. Покажем, что последовательность $\{x^{(k)}\}$ имеет предел. Это имеет место, если выполнен критерий Коши

$$\|x^{(n)}-x^{(m)}\| \to 0 \text{ при } n,\ m \to 0. \tag{9.2.6}$$

Положим для определенности $n > m$ и оценим $\|x^{(n)}-x^{(m)}\|$. Имеем (в силу сжимаемости φ)

$$\begin{aligned}\|x^{(n)}-x^{(m)}\| &= \|\varphi(x^{(n-1)})-\varphi(x^{(m-1)})\| \leqslant \alpha\|x^{(n-1)}-x^{(m-1)}\| = \\ &= \alpha\|\varphi(x^{(n-2)})-\varphi(x^{(m-2)})\| \leqslant \alpha^2\|x^{(n-2)}-x^{(m-2)}\| = \\ &= \ldots \leqslant \alpha^m\|x^{(n-m)}-x^{(0)}\|.\end{aligned}$$

Отсюда следует, что

$$\|x^{(n)}-x^{(m)}\| \leqslant \alpha^m\|x^{(n-m)}-x^{(0)}\|.$$

Оценим $\|x^{(n-m)}-x^{(0)}\|$. Имеем следующую цепочку соотношений:

$$\begin{aligned}\|x^{(n-m)}-x^{(0)}\| &= \|x^{(1)}-x^{(0)}+x^{(2)}-x^{(1)}+x^{(3)}-x^{(2)}+\ldots+ \\ &+x^{(n-m)}-x^{(n-m-1)}\| \leqslant \|x^{(0)}-x^{(1)}\|+\|x^{(1)}-x^{(2)}\|+\ldots+ \\ &+\|x^{(n-m-1)}-x^{(n-m)}\| \leqslant \|x^{(0)}-x^{(1)}\|\left(1+\alpha+\alpha^2+\ldots+\alpha^{n-m-1}\right) \leqslant \\ &\leqslant \frac{\|x^{(0)}-x^{(1)}\|}{1-\alpha}.\end{aligned}$$

Отсюда следует, что

$$\|x^{(n)}-x^{(m)}\| \leqslant \frac{\alpha^m}{1-\alpha}\|x^{(0)}-x^{(1)}\|.$$

Так как $0<\alpha<1$, то при $m \to \infty$ получаем

$$\|x^{(n)}-x^{(m)}\| \to 0,$$

т. е. выполняется критерий Коши (9.2.6), а следовательно, последовательность $\{x^{(k)}\}$ имеет предел при любом начальном $x^{(0)}$:

$$x = \lim_{k\to\infty} x^{(k)}.$$

Можно показать, что сжимающее отображение $\varphi(x)$ является непрерывным. Тогда, переходя в (9.2.5) к пределу, получим, что предел x — решение уравнения (9.2.2). Таким образом, существование решения доказано.

Единственность решения докажем от противного. Предположим, что существует два решения $\bar{x}$ и $\bar{\bar{x}}$, т. е.

$$\bar{x}=\varphi(\bar{x}),\quad \bar{\bar{x}}=\varphi(\bar{\bar{x}}),$$

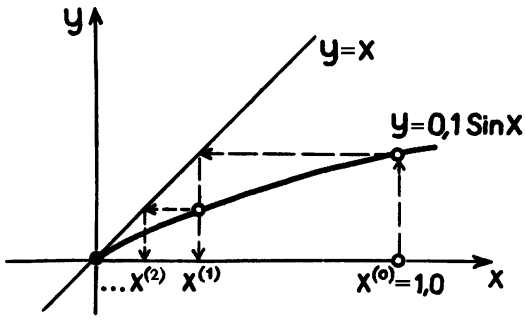

Рис. 9.8

при этом $\|\bar{x}-\bar{\bar{x}}\| \neq 0$. С одной стороны, имеем

$$\|\varphi(\bar{x})-\varphi(\bar{\bar{x}})\| = \|\bar{x}-\bar{\bar{x}}\|.$$

С другой стороны, в силу сжимаемости $\varphi(x)$

$$\|\varphi(\bar{x})-\varphi(\bar{x})\| \leqslant \alpha \|\bar{x}-\bar{\bar{x}}\|.$$

Из последних двух соотношений находим

$$\|\bar{x}-\bar{\bar{x}}\| \leqslant \alpha \|\bar{x}-\bar{x}\|,$$

что при значениях $0<\alpha<1$ и $\|\bar{x}-\bar{\bar{x}}\| \neq 0$ невозможно. Теорема доказана полностью.

Для иллюстрации теоремы рассмотрим два примера.

1) Уравнение

$$x = 0{,}1 \sin x$$

имеет в качестве $\varphi(x)$ сжимающее E^1 в себя отображение

$$\varphi(x) = 0{,}1 \sin x.$$

Единственное решение этого уравнения $x=0$ получается как предел последовательности $\{x^{(k)}\}$ с любым начальным значением $x^{(0)}$:

$$x^{(k+1)} = 0{,}1 \sin x^{(k)}, \quad k = 0, 1, 2, \ldots.$$

Пусть

$$x^{(0)} = 1{,}0,$$

$$x^{(1)} = 0{,}1 \sin 1{,}0, \quad x^{(2)} = 0{,}1 \sin(0{,}1 \sin 1{,}0), \ldots.$$

Ход итерационного процесса изображен на рис. 9.8.

2) Уравнение

$$x = \frac{\pi}{2} \sin x$$

имеет в качестве $\varphi(x)$ несжимающее в E^1 отображение $\varphi(x) = \pi/2 \sin x$. Это уравнение имеет три решения: $x=0$, $\pi/2$, $-\pi/2$ в E^1 (рис. 9.9). Причем, как бы ни было близко к нулю начальное приближение $x^{(0)}$, последовательные приближения не сходятся к корню $x=0$. Если $x^{(0)}>0$, то $\lim\limits_{k\to\infty} x^{(k)} = \pi/2$, если $x^{(0)}<0$, то $\lim\limits_{k\to\infty} x^{(k)} = -\pi/2$ (рис. 9.9).

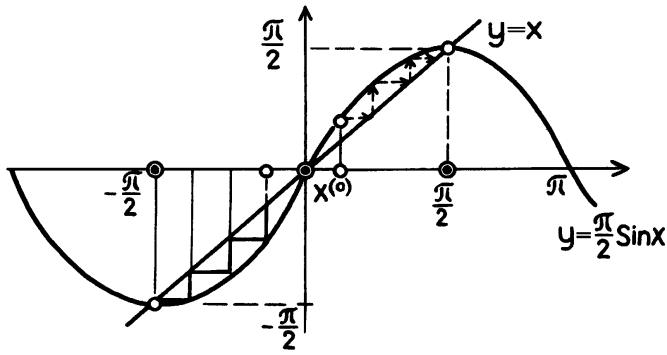

Рис. 9.9

Последний пример показывает, что сходимость последовательных приближений метода простых итераций к решению может быть и тогда, когда $\varphi(x)$ — несжимающее отображение во всем пространстве E^n. Справедливо следующее утверждение.

Теорема 9.3. *Пусть* $\varphi(x)$ — *сжимающее отображение шара* S *пространства* E^n:

$$S=\{x:\ \|x-x^{(0)}\| \leqslant r\}.$$

Пусть также

$$\|\varphi(x^{(0)})-x^{(0)}\| \leqslant (1-\alpha)r. \tag{9.2.7}$$

Тогда в шаре S *существует единственное решение уравнения* (9.2.3), *к которому сходится последовательность* $\{x^{(k)}\}$, *получаемая методом простой итерации* (9.2.5) *с начальным элементом* $x^{(0)}$.

Условие (9.2.7) достаточно для того, чтобы $\varphi(x)$ отображало шар S в себя.

9.2.3. Оценка погрешности метода простой итерации. Доказательство следующих ниже оценок аналогично соответствующим доказательствам для метода простой итерации в линейных системах уравнений (см. **8.3**). Пусть выполнены условия теоремы 9.3, тогда справедливы оценки для нормы разности k-го приближения $x^{(k)}$, полученного из (9.2.5), и точного решения x уравнения $x=\varphi(x)$.

Априорная оценка погрешности

$$\|x^{(k)}-x\| \leqslant \alpha^k r.$$

Апостериорная оценка погрешности

$$\|x^{(k)}-x\| \leqslant \frac{\alpha}{1-\alpha}\|x^{(k)}-x^{(k-1)}\|.$$

Если задана абсолютная точность вычислений ε, то итерационный процесс прекращается по достижении такого значения k, при котором правая часть приведенных выше неравенств становится $\leqslant\varepsilon$. Из априорной оценки можно найти требуемое число итераций

$$\alpha^k r=\varepsilon,\ \ k=\ln(\varepsilon/r)/\ln\alpha,$$

для достижения заданной точности ε.

9.2.4. Метод простой итерации для скалярных и векторных функций $\varphi(x)$. Выше уже приводились примеры отображений $\varphi(x)$, определяемых скалярными функциями $\varphi(x)$. В настоящем разделе приводятся достаточные условия, которым должны удовлетворять скалярные и векторные функции $\varphi_i(x_1, \ldots, x_n)$, $1\leqslant i\leqslant n$, чтобы они определяли сжимающие в S отображения.

Рассмотрим нелинейное уравнение

$$x=\varphi(x),$$

где $\varphi(x)$ — скалярная функция. Норму в E определим $\|x\| = |x|$, шар $S = \{x: |x - x^{(0)}| \leqslant r\}$.

Теорема 9.4. *Пусть функция* $\varphi(x)$ *непрерывно дифференцируема в шаре* S. *Пусть в шаре* S

$$\left|\frac{d\varphi}{dx}(x)\right| \leqslant \alpha < 1 \tag{9.2.8}$$

и

$$|\varphi(x^{(0)}) - x^{(0)}| \leqslant (1-\alpha)r. \tag{9.2.9}$$

Тогда справедливо утверждение теоремы 9.3.

Доказательство. Утверждение будет следовать из теоремы 9.3, если мы покажем, что выполняются ее условия. Очевидно, что (9.2.7) следует из (9.2.9). По теореме о среднем для $\bar{x}$, $\bar{\bar{x}} \in S$ имеем

$$\varphi(\bar{x}) - \varphi(\bar{\bar{x}}) = \frac{d\varphi}{dx}(\xi)(\bar{x} - \bar{\bar{x}}),$$

где $\xi \in S$. Отсюда следует, что

$$|\varphi(\bar{x}) - \varphi(\bar{\bar{x}})| = \left|\frac{d\varphi}{dx}(\xi)\right| |\bar{x} - \bar{\bar{x}}| \leqslant \alpha |x - \bar{\bar{x}}|,$$

а так как из (9.2.8) имеем $\alpha < 1$, то отображение $\varphi(x)$ — сжимающее в шаре S. Все условия теоремы 9.3 выполнены, что и требовалось показать.

Для иллюстрации этой теоремы рассмотрим определение положительного корня уравнения

$$x = \frac{\pi}{2}\sin x$$

методом простой итерации

$$x^{(k+1)} = \frac{\pi}{2}\sin x^{(k)}, \quad k = 0, 1, 2, \ldots; \quad x^{(0)} = 1{,}5$$

в шаре $S = \{x: |x - 1{,}5| \leqslant 0{,}1\}$. Определим коэффициент сжатия:

$$\alpha = \max_{1{,}4 \leqslant x \leqslant 1{,}6} \left|\frac{\pi}{2}\cos x\right| = 0{,}267.$$

Проверим условие (9.2.9):

$$\left|\frac{\pi}{2}\sin 1{,}5 - 1{,}5\right| \cong 0{,}0667 \leqslant (1 - 0{,}267)0{,}1 = 0{,}0733.$$

Таким образом, оно выполнено. Согласно утверждению теоремы 9.4, последовательность $\{x^{(k)}\}$ сходится к единственному решению уравнения в S. Вычислим два первых приближения:

$$x^{(1)} = \frac{\pi}{2}\sin 1{,}5 = 1{,}56686; \quad x^{(2)} = \frac{\pi}{2}\sin 1{,}56686 = 1{,}57078.$$

Точное значение корня $x=\pi/2$ с шестью верными знаками после запятой 1,570796 показывает, что $x^{(2)}$ дает четыре верных знака после запятой. Ожидаемая погрешность второго приближения, следующая из априорной оценки

$$|x^{(2)}-x| \leqslant (0{,}267)^2 \cdot 0{,}1 \cong 0{,}007,$$

оказывается более пессимистической. Это обычное явление, поскольку при получении оценок берутся завышенные значения в правых частях неравенств. Из апостериорной оценки находим неравенство

$$|x^{(2)}-x| \leqslant \frac{0{,}267}{1-0{,}267} 3{,}98 \cdot 10^{-3} \cong 0{,}001,$$

означающее, что в $x^{(2)}$ по крайней мере два верных знака после запятой.

Обычно точное значение корня неизвестно, а единственная информация о погрешности может быть получена только из оценок, поэтому следовало бы продолжить итерации дальше, чтобы гарантировать четыре верных знака после запятой у приближенного решения $x^{(k)}$.

Рассмотрим нелинейное уравнение $x=\varphi(x)$, где φ — непрерывно дифференцируемая векторная функция, т. е. рассмотрим систему n уравнений с n неизвестными

$$\begin{aligned}
x_1 &= \varphi_1(x_1, \ldots, x_n),\\
x_2 &= \varphi_2(x_1, \ldots, x_n),\\
&\ldots\ldots\ldots\ldots\ldots\ldots\\
x_n &= \varphi_n(x_1, \ldots, x_n).
\end{aligned}$$

Зададим в E^n норму x (например, $\| x \| = \max\limits_{1 \leqslant i \leqslant n} |x_i|$), шар $S=\{x: \| x- -x^{(0)} \| \leqslant r\}$, определим матрицу:

$$B(x)=\begin{pmatrix}
\dfrac{\partial \varphi_1}{\partial x_1}(x_1, \ldots, x_n) \ldots \dfrac{\partial \varphi_1}{\partial x_n}(x_1, \ldots, x_n)\\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
\dfrac{\partial \varphi_n}{\partial x_1}(x_1, \ldots, x_n) \ldots \dfrac{\partial \varphi_n}{\partial x_n}(x_1, \ldots, x_n)
\end{pmatrix}.$$

Верно следующее утверждение.

Теорема 9.5. *Пусть функция $\varphi(x)$ непрерывно дифференцируема в шаре S. Пусть также в шаре S*

$$\| B(x) \| \leqslant \alpha < 1$$

и

$$\| \varphi(x^{(0)}) - x^{(0)} \| \leqslant (1-\alpha) r.$$

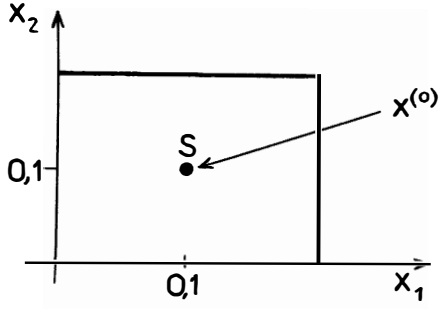

Рис. 9.10

Тогда справедливо утверждение теоремы 9.3.

Для иллюстрации теоремы используем метод простой итерации для приближенного решения системы уравнений

$$x_1 = 0{,}05e^{-x_1x_2},$$
$$x_2 = 0{,}05e^{-(x_1+x_2)}.$$

Определим шар $S = \max\limits_{1\leqslant i\leqslant 2} |x_i - 0{,}1| \leqslant 0{,}1$ (рис. 9.10), а также матрицу

$$B(x) = \begin{pmatrix} -0{,}05x_2e^{-x_1x_2} - 0{,}05x_1e^{-x_1x_2} \\ -0{,}05e^{-(x_1+x_2)} - 0{,}05e^{-(x_1+x_2)} \end{pmatrix}.$$

Вычислим коэффициент сжатия:

$$\alpha = \max_S \left(0{,}05(x_1+x_2)e^{-x_1x_2},\ 0{,}1e^{-(x_1-x_2)}\right) = 0{,}1.$$

Проверим условие (9.2.7), имеем

$$\| \varphi(x^{(0)}) - x^{(0)} \| = \max\left(|0{,}05e^{-(0{,}1)^2} - 0{,}1|,\ |0{,}05e^{-0{,}2} - 0{,}1|\right) =$$
$$= \max\left(5\cdot 10^{-2},\ 6\cdot 10^{-2}\right) < (1-\alpha)r = 9\cdot 10^{-2}.$$

Условие выполнено. По теореме 9.5 последовательные приближения $\{x^{(k)}\}$ сходятся к точному решению

$$x_1^{(k+1)} = 0{,}05e^{-x_1^{(k)}x_2^{(k)}}; \qquad x_1^{(0)} = 0{,}1$$
$$x_2^{(k+1)} = 0{,}05e^{-(x_1^{(k)}+x_2^{(k)})}; \qquad x_2^{(0)} = 0{,}1;\ k = 0,\ 1,\ 2,\ \ldots.$$

Вычислим два первых приближения:

$x_1^{(1)} = 0{,}05e^{-(0{,}1)^2} = 4{,}950249\cdot 10^{-2}$; $x_1^{(2)} = 0{,}05e^{-x_1^{(1)}x_2^{(1)}} = 4{,}989878\cdot 10^{-2}$, ...

$x_2^{(1)} = 0{,}05e^{-(0{,}2)} = 4{,}093654\cdot 10^{-2}$; $x_2^{(2)} = 0{,}05e^{-(x_1^{(1)}+x_2^{(1)})} = 4{,}56765\cdot 10^{-2}$.

●9.3. Метод Ньютона

В основе метода Ньютона, как уже отмечалось в **9.1,** лежит линеаризация нелинейного уравнения. Рассмотрим сначала метод Ньютона для скалярных уравнений, а затем для векторных.

9.3.1. Метод Ньютона для скалярных уравнений. Пусть имеется нелинейное уравнение

$$f(x) = 0, \tag{9.3.1}$$

которое будем рассматривать в $S = \{x : |x - x^{(0)}| \leqslant r\}$. Предположим, что $f'(x) \neq 0$ в S. В точке $x^{(0)}$ произведем линеаризацию (9.3.1), т. е. заменим в (9.3.1) функцию $f(x)$ линейной функцией, проходящей

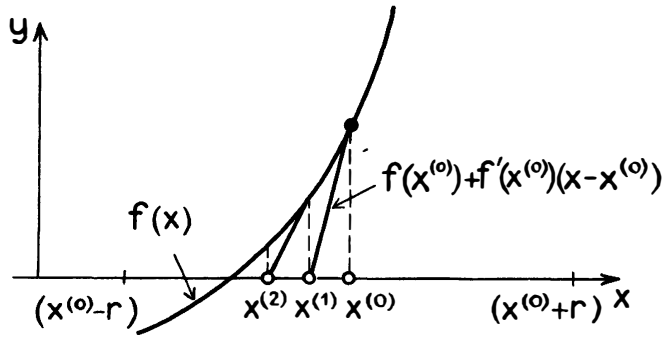

Рис. 9.11

через $x^{(0)}$ и имеющей производную, равную $f'(x^{(0)})$ (рис. 9.11). Получим линеаризованное уравнение

$$f(x^{(0)})+f'(x^{(0)})(x-x^{(0)})=0. \tag{9.3.2}$$

Найдем корень уравнения (9.3.2). Обозначая его $x^{(1)}$, имеем

$$x^{(1)}=x^{(0)}-\frac{f(x^{(0)})}{f'(x^{(0)})}. \tag{9.3.3}$$

Теперь произведем линеаризацию (9.3.1) в точке $x^{(1)}$, повторим все этапы получения формулы (9.3.3), заменяя при этом $x^{(0)}$ на $x^{(1)}$, а $x^{(1)}$ — на $x^{(2)}$ (см. рис. 9.11). Таким образом, для произвольного k получим формулу

$$x^{(k+1)}=x^{(k)}-\frac{f(x^{(k)})}{f'(x^{(k)})},\quad k=0,\ 1,\ 2,\ \ldots, \tag{9.3.4}$$

метода Ньютона приближенного определения корня уравнения $f(x)=0$.

Метод Ньютона называют еще *методом касательных*, что вполне соответствует геометрическому смыслу линеаризации на одном шаге.

Метод Ньютона является методом простой итерации

$$x^{(k+1)}=\varphi(x^{(k)}),\quad k=0,\ 1,\ 2,\ \ldots,$$

со специально подобранной по $f(x)$ функцией $\varphi(x)$, а именно

$$\varphi(x)=x-\frac{f(x)}{f'(x)};$$

исходное уравнение при $f'(x)\neq 0$ эквивалентно уравнению

$$x=\varphi(x).$$

Следующий пример показывает, что метод Ньютона может не давать последовательность $\{x^{(k)}\}$, сходящуюся к корню (нарушено условие сходимости). Уравнение

$$\operatorname{arctg} x=0$$

имеет единственный корень $x=0$. Итерационный процесс Ньютона (9.3.4) примет вид

$$x^{(k+1)}=x^{(k)}-(1+(x^{(k)})^2\operatorname{arctg} x^{(k)} \tag{9.3.5}$$

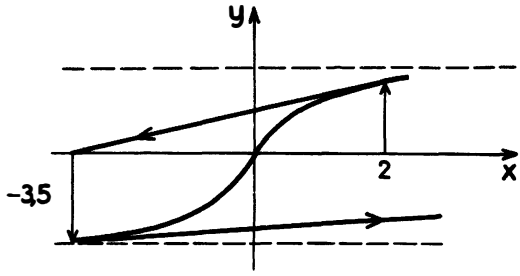

Рис. 9.12

и для достаточно больших значений $|x^{(0)}|$ из (9.3.5) получаем расходящуюся последовательность $\{x^{(k)}\}$. Например, $x^{(0)}=2$, $x^{(1)}=-3{,}5$, $x^{(2)}=13{,}9$, $x^{(3)}=-279$ (рис. 9.12). В то же время для малых значений $|x^{(0)}|$ из (9.3.5) получаем сходящуюся к корню последовательность $\{x^{(k)}\}$. Например, $x^{(0)}=1$, $x^{(1)}=-0{,}57$, $x^{(2)}=0{,}17$, $x^{(3)}=-10^{-3}$, $x^{(4)}=9\cdot 10^{-8}$.

9.3.2. Достаточные условия сходимости метода Ньютона. Выведем достаточные условия сходимости метода Ньютона в предположении, что $f(x)$ дважды непрерывно дифференцируема, из соответствующих условий метода простой итерации (см. теорему 9.4). Условие (9.2.8), учитывая вид $\varphi(x)$, запишем так:

$$\left|\frac{d\varphi}{dx}\right|=\frac{|f(x)f''(x)|}{(f'(x))^2}\leqslant\alpha<1. \tag{9.3.6}$$

Условие (9.2.9) принимает вид

$$|\varphi(x^{(0)})-x^{(0)}|=\left|\frac{f(x^{(0)})}{f'(x^{(0)})}\right|\leqslant(1-\alpha)r. \tag{9.3.7}$$

Окончательно, если в области $S=\{x:|x-x^{(0)}|\leqslant r\}$ выполнены условия (9.3.6), (9.3.7), последовательность $\{x^{(k)}\}$, получаемая методом Ньютона (9.3.4), сходится к решению уравнения $f(x)=0$ в S.

Скорость сходимости $x^{(k)}$ к решению находится из априорной оценки метода простой итерации $|x^{(k)}-x|\leqslant\alpha^k r$. Однако эта оценка для метода Ньютона оказывается слишком грубой, при приближении $x^{(k)}$ к корню скорость сходимости является квадратичной.

Этот факт вытекает из следующих достаточных условий сходимости метода Ньютона, которые приводятся без доказательства.

Определим числа:

$$p_0=\left|\frac{f(x^{(0)})}{f'(x^{(0)})}\right|,\quad p_1\geqslant\max_s\left|\frac{f''(x)}{f'(x)}\right|,$$

$$h=p_0p_1,\quad r=(1-\sqrt{1-2h})p_0h^{-1}.$$

Если

$$\alpha=2h<1, \tag{9.3.8}$$

то последовательность $\{x^{(k)}\}$ метода Ньютона сходится к корню x уравнения $f(x)=0$ в области S и имеет место оценка погрешности

$$|x^{(k)}-x|\leqslant\alpha^{2^k}2^{-k}p_0h^{-1},\quad k=0,\ 1,\ 2,\ \ldots.$$

9.3.3. Применение программы B4A1. Для определения корня скалярного уравнения $f(x)=0$ методом Ньютона может служить

библиотечная программа B4A1. Например, для уравнения $\operatorname{arctg} x = 0$ поиск корня с точностью ε (смысл ε поясняется в описании B4A1) можно выполнить с помощью следующей программы:

```
      REAL X,R,D,X0,E
      INTEGER N,I
      EXTERNAL F
C     ВВОД НАЧАЛЬНОГО ПРИБЛИЖЕНИЯ К КОРНЮ X0
C     ВВОД ТОЧНОСТИ E, ВВОД МАКСИМАЛЬНОГО ЧИС-
C     ЛА ИТЕРАЦИЙ N
      READ (5,1) X0,E,N
1     FORMAT (2F9.6,I5)
C     ОБРАЩЕНИЕ К ПРОГРАММЕ B4A1
      CALL B4A1(X,R,D,F,X0,E,N,I)
C     ВЫВОД НА ТЕРМИНАЛ ЗНАЧЕНИЯ КОРНЯ X И
C     ИНДЕКСА ОШИБОК I
      WRITE (5,2) X,I
2     FORMAT (2X,'X=',E13.6,'I=',I2)
      END
C     ПОДПРОГРАММА F, ВЫЧИСЛЯЮЩАЯ ЗНАЧЕНИЕ
C     ФУНКЦИИ И ЕЕ ПРОИЗВОДНОЙ
      SUBROUTINE F(X,R,D)
      REAL X,R,D
      R=ATAN(X)
      D=1./(1.+X*X)
      RETURN
      END
```

9.3.4. Метод Ньютона для векторного уравнения. Пусть имеется векторное уравнение

$$f(x)=0,$$

которое эквивалентно системе n скалярных уравнений

$$\begin{aligned} &f_1(x_1, x_2, \ldots, x_n)=0,\\ &f_2(x_1, x_2, \ldots, x_n)=0,\\ &\ldots\ldots\ldots\ldots\ldots\ldots\\ &f_n(x_1, x_2, \ldots, x_n)=0. \end{aligned} \qquad (9.3.9)$$

Будем рассматривать (9.3.9) в шаре $S=\{x: \| x-x^{(0)} \| \leqslant r\}$. Предположим, что в любой точке $x \in S$ матрица $A(x)$ с элементами

$$A_{i,j}(x)=\frac{\partial f_i}{dx_j}(x_1, \ldots, x_n)$$

невырождена.

Тогда метод Ньютона для (9.3.9) по аналогии с (9.3.4) записывается следующим образом:

$$x^{(k+1)}=x^{(k)}-A^{-1}(x^{(k)})f(x^{(k)}), \quad k=0, 1, \ldots. \qquad (9.3.10)$$

На каждом шаге k итерационного процесса (9.3.10) решаются линеаризованные уравнения, полученные из (9.3.9).

Условия сходимости последовательности $\{x^{(k)}\}$ к решению (9.3.9) можно получить из теоремы 9.5, рассматривая (9.3.10) как метод простой итерации с

$$\varphi(x)=Ex-A^{-1}(x)f(x),$$
$$x^{(k+1)}=\varphi(x^{(k)}).$$

Здесь E — единичная матрица. При этом следует потребовать, чтобы функции $f_i(x_1, \ldots, x_n)$ были дважды непрерывно дифференцируемы по своим аргументам в шаре S.

9.3.5. Применение программы B4A2. Для определения корней системы уравнений (9.3.9) методом Ньютона может использоваться библиотечная программа B4A2. Например, пусть имеем систему уравнений

$$x_1-0{,}05e^{-x_1x_2}=0,$$
$$x_2-0{,}05e^{-(x_1+x_2)}=0.$$

Точность вычислений ε определена в описании B4A2. Поиск приближенного значения корня $x=(x_1, x_2)$ с точностью ε можно выполнить с помощью следующей программы:

```
      REAL X(2),F(2),E,W(10)
      INTEGER N,K,I
      EXTERNAL R
      DATA I/0/,N/2,K/10/
C     ВВОД НАЧАЛЬНОГО ПРИБЛИЖЕНИЯ К КОРНЮ
C     ВВОД ТОЧНОСТИ E
      READ (5.1) X,E
1     FORMAT (3F9.6)
C     ОБРАЩЕНИЕ К ПРОГРАММЕ B4A2
      CALL B4A2(R,N,X,F,E,W,K,I)
C     ВЫВОД НА ТЕРМИНАЛ ЗНАЧЕНИЙ КОМПОНЕНТ
C     КОРНЯ И ИНДЕКСА ОШИБОК I
      WRITE (5.2) X,I
2     FORMAT (2X,2E13.6,'I-',I2)
      END
C     ПОДПРОГРАММА, ВЫЧИСЛЯЮЩАЯ ЗНАЧЕНИЯ
C     ФУНКЦИЙ
      SUBROUTINE R(N,X,F,J)
      REAL X(2),F(2)
      INTEGER N,J
      DATA J/0/
      F(1)=X(1)-0.05*EXP(-X(1)*X(2))
      F(2)=X(2)-0.05*EXP(-(X(1)+X(2)))
      RETURN
      END
```

В программе B4A2 необходимые значения производных $\frac{\partial f_i}{\partial x_j}$ для определения матрицы A вычисляются с помощью формул численного дифференцирования (см. гл. 10).

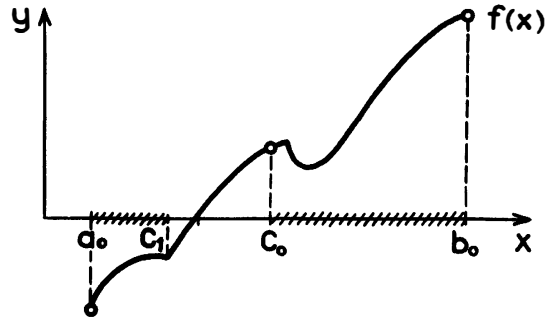

Рис. 9.13

● 9.4. Метод бисекции

Метод бисекции (или метод деления отрезка пополам) отличается от рассмотренных выше способов приближенного определения корня уравнения $f(x)=0$ тем, что для его сходимости не требуется, чтобы $f(x)$ была дифференцируемой, достаточно только непрерывности $f(x)$.

9.4.1. Алгоритм метода бисекции. Пусть задана непрерывная на интервале $[a_0, b_0]$ функция $f(x)$. Пусть известно, что на $[a_0, b_0]$ имеется единственный корень уравнения

$$f(x)=0. \tag{9.4.1}$$

Тогда в силу непрерывности $f(x)$ значения $f(a_0)$ и $f(b_0)$ имеют разные знаки (рис. 9.13). Разделим отрезок $[a_0, b_0]$ пополам точкой $c_0=(a_0+b_0)/2$. Сохраняем для дальнейшего деления ту половину отрезка, на концах которого $f(x)$ имеет разные знаки. Введем следующие обозначения:

$$a_1=\begin{cases} c_0, \text{ если } f(a_0)f(c_0)>0, \\ a_0, \text{ если } f(a_0)f(c_0)<0; \end{cases}$$

$$b_1=\begin{cases} c_0, \text{ если } f(b_0)f(c_0)>0, \\ b_0, \text{ если } f(b_0)f(c_0)<0. \end{cases}$$

Разделим оставшийся отрезок $[a_1, b_1]$ пополам и повторим приведенную систему обозначений. Для произвольного k-го деления имеем

$$c_k=\frac{a_k+b_k}{2}, \quad k=1, 2, \ldots$$

$$a_{k+1}=\begin{cases} c_k, \text{ если } f(a_k)f(c_k)>0, \\ a_k, \text{ если } f(a_k)f(c_k)<0; \end{cases}$$

$$b_{k+1}=\begin{cases} c_k, \text{ если } f(b_k)f(c_k)>0, \\ b_k, \text{ если } f(b_k)f(c_k)<0. \end{cases}$$

Алгоритм завершается, если для какого-либо k оказывается $f(c_k)=0$. Тогда значение корня $x=c_k$. Если задана абсолютная точность определения корня ε, то алгоритм завершается при условии

$$\frac{b_k-a_k}{2}=\frac{b_0-a_0}{2^{k+1}}<\varepsilon.$$

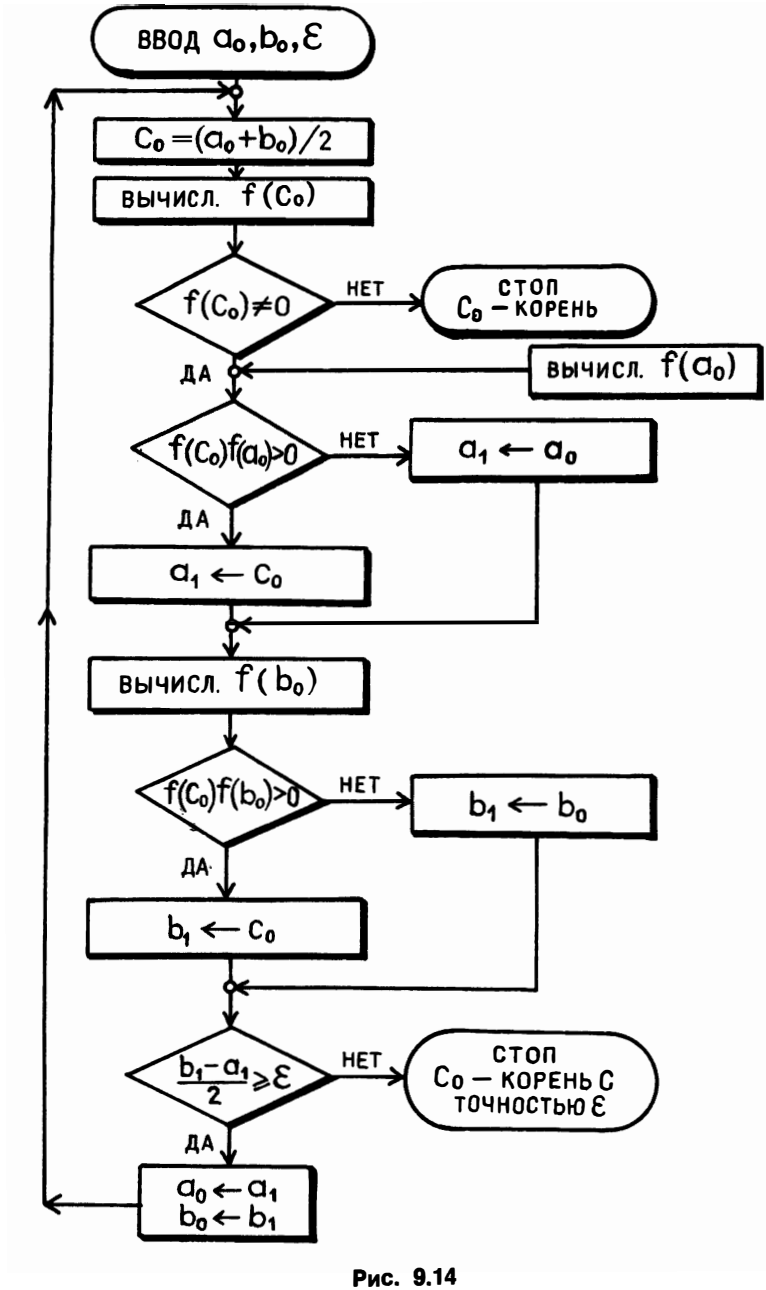

Рис. 9.14

В этом случае за приближенное значение корня x принимается значение c_k; очевидно, что

$$|x - c_k| \leqslant \frac{b_0 - a_0}{2^{k+1}} < \varepsilon.$$

9.4.2. Схема алгоритма бисекции. Отличительной особенностью алгоритма метода бисекции является его логическая структура.

В блок-схеме, представленной на рис. 9.14, имеется четыре логических элемента типа альтернативы (см. гл. 2). Концы интервала до деления пополам обозначаются a_0, b_0, после деления — a_1, b_1.

9.4.3. Применение библиотечной программы B4A0. Для определения корня непрерывной функции $f(x)$ методом бисекции можно использовать программу B4A0, которая реализует схему алгоритма, приведенного на рис. 9.17, с ограничением на допустимое число делений пополам исходного интервала $[a, b]$. Пусть ищется корень уравнения

$$x - e^{-x} = 0$$

на интервале [0, 1] с точностью $\varepsilon = 10^{-5}$, максимальное число делений интервала пополам положим равным 20. Тогда программа может иметь следующий вид:

```
      REAL X,R,A,B,E
      INTEGER N,I
      EXTERNAL F
      DATA A,B,E,N/0.,1.,1.E-5,20/
C     ОБРАЩЕНИЕ К ПРОГРАММЕ B4A0
      CALL B4A0(X,R,F,A,B,E,N,I)
C     ВЫВОД НА ТЕРМИНАЛ ПРИБЛИЖЕННОГО
C     ЗНАЧЕНИЯ КОРНЯ И ИНДЕКСА ОШИБОК
      WRITE (5,1) X,I
1     FORMAT (2X,'X=',E13.6,'I=',I2)
      END
C     ФУНКЦИЯ-ПОДПРОГРАММА, ВЫЧИСЛЯЮЩАЯ
C     ЗНАЧЕНИЕ F(X)
      FUNCTION F(X)
      REAL X
      F=X-EXP(-X)
      RETURN
      END
```

● 9.5. Поиск минимума функции

Поиск оптимальных инженерно-технических, экономических и научных решений — основное поле деятельности инженера и ученого. Задачи минимизации функций одного или нескольких переменных принадлежат к этому важному направлению.

Функция $\Phi(x_1, \ldots, x_n)$, подлежащая минимизации, называется *целевой* функцией.

Задача поиска максимума $\Phi(x_1, \ldots, x_n)$ сводится к минимизации функции $-\Phi(x_1, \ldots, x_n)$, поэтому, не ограничивая общности, будем рассматривать задачу поиска минимума.

Примерами целевых функций могут служить: расход топлива Ф, необходимый для вывода космического аппарата на орбиту (за счет выбора параметров аппарата и траектории $x = (x_1, \ldots, x_n)$

стараются минимизировать величину Ф); вес Ф механической конструкции с заданной прочностью; стоимость Ф выпускаемой продукции предприятия и т. п.

Наибольшие трудности поиска минимума $\Phi(x)$ возникают, когда размерность n вектора x велика. Поэтому важнейшей является проблема уменьшения размерности вектора целевой функции на этапе построения математической модели технической задачи. В модели следует сохранить только те переменные x_i, которые сильнее других влияют на изменение целевой функции. Естественно, что такая информация обычно связана с опытом и знаниями в конкретных предметных областях.

9.5.1. Решение уравнений и минимизация. Покажем, как поиск корня системы нелинейных уравнений можно свести к задаче минимизации и, наоборот, минимизацию — к решению системы уравнений.

Предположим, что в области D пространства E^n необходимо решить систему уравнений

$$\begin{aligned} &f_1(x_1, x_2, \ldots, x_n)=0,\\ &f_2(x_1, x_2, \ldots, x_n)=0,\\ &\ldots\ldots\ldots\ldots\ldots\ldots\\ &f_n(x_1, x_2, \ldots, x_n)=0. \end{aligned} \tag{9.5.1}$$

Пусть известно, что в D существует решение (9.5.1). Определим целевую функцию $\Phi(x)$ формулой

$$\Phi(x_1, \ldots, x_n)=f_1^2(x_1, \ldots, x_n)+f_2^2(x_1, \ldots, x_n)+\ldots+f_n^2(x_1, \ldots, x_n). \tag{9.5.2}$$

В области D целевая функция $\Phi(x)\geqslant 0$. Минимальное значение $\Phi(x)$ достигает на корне (9.5.1) — векторе $x=(x_1, \ldots, x_n)$, который обращает все $f_i(x)$ в нуль. Поэтому решение системы (9.5.1) эквивалентно поиску минимума Ф, определенной формулой (9.5.2): найти

$$\min_D \Phi(x). \tag{9.5.3}$$

Если минимальное значение $\Phi(x)$ в D строго больше нуля, то это свидетельствует, что в D система уравнений (9.5.1) не имеет решений.

Теперь рассмотрим задачу (9.5.3) как исходную с функцией $\Phi(x)$, непрерывно дифференцируемой в открытой области D. Пусть известно, что в D существует x, доставляющий минимум $\Phi(x)$, тогда в этой точке $x=(x_1, \ldots, x_n)$ с необходимостью должны выполняться соотношения

$$\frac{\partial \Phi}{\partial x_1}(x_1, \ldots, x_n)=0,$$
$$\frac{\partial \Phi}{\partial x_2}(x_1, \ldots, x_n)=0, \qquad (9.5.4)$$
$$\ldots\ldots\ldots\ldots\ldots$$
$$\frac{\partial \Phi}{\partial x_n}(x_1, \ldots, x_n)=0.$$

Система уравнений (9.5.4) имеет форму (9.5.1). Решение (9.5.4) позволяет определить неизвестные экстремальные точки. Затем, используя достаточные условия минимума, следует выделить точки, в которых достигается локальный минимум, затем, сравнив значение целевой функции в точках локального минимума, надо выбрать точку (одну или несколько) с минимальной величиной локального минимума, т. е. определить глобальный минимум $\Phi(x)$ в области D.

В качестве первого примера рассмотрим систему уравнений

$$\begin{cases} x_1 - 0{,}05e^{-x_1 x_2}=0, \\ x_2 - 0{,}05e^{-(x_1+x_2)}=0 \end{cases}$$

в области $D=\{|x_1-0{,}1|\leqslant 0{,}1,\ |x_2-0{,}1|\leqslant 0{,}1\}$. Решение этой системы эквивалентно поиску минимума $\Phi(x_1, x_2)$:

$$\min_D \Phi(x_1 x_2)=\left[(x_1-0{,}05e^{-x_1 x_2})^2+(x_2-0{,}05e^{-(x_1+x_2)})^2\right].$$

В качестве второго примера найдем в области $D=\{|x_1|<1,\ |x_2|<1\}$ минимум $\Phi(x_1, x_2)$:

$$\min_D \Phi(x_1 x_2)=x_1^4+x_2^4-x_1^2-x_2^2.$$

Система уравнений (9.5.4) принимает вид

$$\begin{cases} 2x_1^3-x_1=0, \\ 2x_2^3-x_2=0. \end{cases}$$

Отсюда получаем точки экстремума с координатами $(x_1^{(1)}=0,\ x_1^{(2)}=1/\sqrt{2},\ x_1^{(3)}=-1/\sqrt{2})(x_2^{(1)}=0,\ x_2^{(2)}=1/\sqrt{2},\ x_2^{(3)}=-1/\sqrt{2})$, всего девять точек. Из них четыре точки дают локальный и одновременно глобальный минимум, а именно: $(1/\sqrt{2},\ 1/\sqrt{2})$; $(-1/\sqrt{2},\ 1/\sqrt{2})$; $(-1/\sqrt{2},\ -1/\sqrt{2})$; $(1/\sqrt{2},\ -1/\sqrt{2})$. Минимальное значение целевой функции в этих точках $\Phi=-1/2$.

9.5.2. Поиск минимума функции перебором. Идея алгоритма перебора крайне проста. Вычислим в конечном числе точек $x^{(k)}$ области D значения $\Phi(x)$ и сравним их между собой. Точку $x^{(*)}$, которая дает минимальное значение целевой функции, назовем *приближенным элементом* x, минимизирующим $\Phi(x)$.

Точность определения точки минимума, причем глобального, зависит от плотности заполнения области D дискретным множеством $x^{(k)}$.

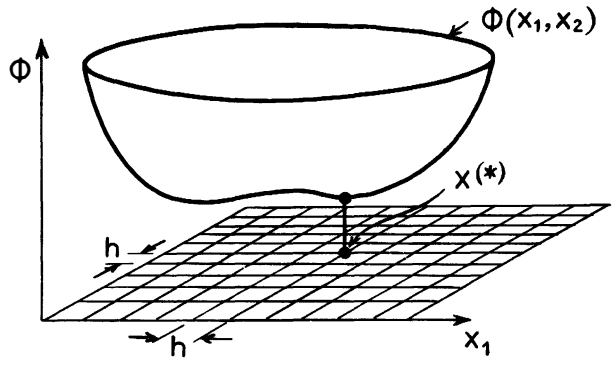

Рис. 9.15

При большой размерности вектора n реализовать на ЭВМ алгоритм перебора невозможно из-за огромного объема вычислительной работы.

Пусть область D — гиперкуб

$$D=\{0\leqslant x_i\leqslant 1,\ \ 1\leqslant i\leqslant n\},$$

в котором ищется $\min\Phi(x)$. Точность определения координат вектора, минимизирующего Ф, положим равной 0,1. Тогда интервалы $0\leqslant x_i\leqslant 1$ следует разбить на 10 частей с шагом $h=0{,}1$ плоскостями, ортогональными x_i (рис. 9.15), и вычислить во всех точках пересечения плоскостей значения $\Phi(x)$. Всего потребуется вычислить $\Phi(x)$ в 10^n точках. Пусть для нахождения Ф в каждой точке требуется, например $\sim 10^2 n$ арифметических операций. Тогда общее число арифметических операций алгоритма перебора $\sim 10^{n+2}n$ и при $n=10$ на ЭВМ с быстродействием 10^6 оп/с требуется 10^7 с, т. е. ~ 4 месяца непрерывной работы ЭВМ.

При небольших размерностях n вектора x перебор является наиболее эффективным методом поиска минимума $\Phi(x)$ в случае сложного поведения $\Phi(x)$ (наличие многих локальных минимумов).

Приведем пример программы поиска минимума перебора

$$\min\Phi(x_1,\ x_2)$$

в области $D=\{0\leqslant x_1\leqslant 1,\ \ 0\leqslant x_2\leqslant 1\}$:

```
      REAL H,X1,X2,F,FM
      INTEGER N
C     ВВОД ЧИСЛА ТОЧЕК ПЕРЕБОРА N ПО КАЖДОЙ
C     ПЕРЕМЕННОЙ
      READ (5,1) N
1     FORMAT (I6)
C     ВЫЧИСЛЕНИЕ ШАГА H ПЕРЕБОРА
      H=1/N
C     АЛГОРИТМ ПЕРЕБОРА
      FM=FI(0.,0.)
      X1=0.
```

```
      X2=0.
      DO 3 I=1,N+1
      DO 3 J=1,N+1
      F=FI(H*(I-1),H*(J-1))
      IF (F.LT.FM) GO TO 2
      GO TO 3
2     FM=F
      X1=H*(I-1)
      X2=H*(J-1)
3     CONTINUE
C     ВЫВОД НА ТЕРМИНАЛ МИНИМАЛЬНОГО
C     ЗНАЧЕНИЯ ФУНКЦИИ FM;
C     ТОЧКИ С КООРДИНАТАМИ X1, X2, В КОТОРОЙ
C     ДОСТИГАЕТСЯ FM
      WRITE (5.4) FM,X1,X2
4     FORMAT (2X,'FM=',E13.6,2X,2E13.6)
      END
C     ФУНКЦИЯ-ПОДПРОГРАММА, ВЫЧИСЛЯЮЩАЯ
C     МИНИМИЗИРУЕМУЮ ФУНКЦИЮ
      FUNCTION FI(X1,X2)
      REAL X1,X2
      FI=...
      RETURN
      END
```

● 9.6. Методы спуска

Идея всех методов спуска состоит в том, чтобы исходя из начального приближения — точки $x^{(0)}=(x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)})\in D$ — перейти в следующую точку $x^{(1)}=(x_1^{(1)}, x_2^{(1)}, \ldots, x_n^{(1)})\in D$ так, чтобы значение $\Phi(x_1^{(0)}, \ldots, x_n^{(0)})$ уменьшилось:

$$\Phi(x^{(1)})<\Phi(x^{(0)}).$$

9.6.1. Метод покоординатного спуска. Пусть в области D задано нулевое приближение

$$x^{(0)}=(x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}).$$

Рассматриваем функцию $\Phi(x_1^{(0)}, x_2^{(0)}, x_3^{(0)}, \ldots, x_n^{(0)})$ при фиксированных значениях $x_2^{(0)}, x_3^{(0)}, \ldots, x_n^{(0)}$ как функцию одной переменной x_1 (рис. 9.16). Находим

$$\min_{x_1\in D}\Phi(x_1, x_2^{(0)}, \ldots, x_n^{(0)}).$$

Значение x_1, доставляющее минимум, обозначаем $x_1^{(1)}$:

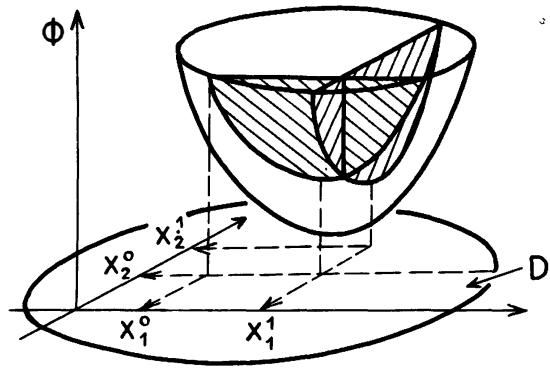

Рис. 9.16

$$\Phi\left(x_1^{(1)}, x_2^{(0)}, \ldots, x_n^{(0)}\right) \leqslant \Phi\left(x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}\right).$$

Далее при фиксированных значениях $x_1^{(1)}, x_3^{(0)}, \ldots, x_n^{(0)}$ рассматриваем $\Phi\left(x_1^{(1)}, x_2, x_3^{(0)}, \ldots, x_n^{(0)}\right)$ как функцию одной переменной x_2. Находим

$$\min_{x_2 \in D} \Phi\left(x_1^{(1)}, x_2, x_3^{(0)}, \ldots, x_n^{(0)}\right).$$

Значение x_2, доставляющее минимум, обозначаем $x_2^{(1)}$, получаем

$$\Phi\left(x_1^{(1)}, x_2^{(1)}, x_3^{(0)}, \ldots, x_n^{(0)}\right) \leqslant \Phi\left(x_1^{(1)}, x_2^{(0)}, x_3^{(0)}, \ldots, x_n^{(0)}\right)$$

и т. д., после n шагов получаем

$$\Phi\left(x_1^{(1)}, \ldots, x_n^{(1)}\right) \leqslant \Phi\left(x_1^{(1)}, x_2^{(1)}, \ldots, x_n^{(0)}\right).$$

В результате одного шага покоординатного спуска происходит переход из начальной точки $x^{(0)} = \left(x_1^{(0)}, \ldots, x_n^{(0)}\right)$ в точку $x^{(1)} = \left(x_1^{(1)}, \ldots, x_n^{(1)}\right)$. Если при этом оказывается, что

$$\Phi\left(x_1^{(1)}, \ldots, x_n^{(1)}\right) = \Phi\left(x_1^{(0)}, \ldots, x_n^{(0)}\right),$$

то начальный элемент $x^{(0)}$ — точка, доставляющая экстремум $\Phi(x)$. Если $\Phi\left(x^{(1)}\right) < \Phi\left(x^{(0)}\right)$, то выполняется следующий шаг покоординатного спуска, в котором за начальную точку принимается $x^{(1)}$, получаем $x^{(2)}$ и т. д. Этот процесс продолжается до тех пор, пока не выполнится какое-либо условие окончания алгоритма, например такое:

$$\left|\Phi\left(x^{(k+1)}\right) - \Phi\left(x^{(k)}\right)\right| < \varepsilon, \tag{9.6.1}$$

где ε — заданная точность.

Заметим, что центральным звеном рассматриваемого алгоритма является поиск минимума функции одной переменной (см. **9.7**).

9.6.2. Метод градиентного спуска. Напомним, что градиент функции $\Phi(x)$ определяется формулой

$$\operatorname{grad} \Phi(x) = \left(\frac{\partial \Phi}{\partial x_1}(x), \frac{\partial \Phi}{\partial x_2}(x), \ldots, \frac{\partial \Phi}{\partial x_n}(x)\right).$$

Вектор $\operatorname{grad} \Phi(x)$ ортогонален линиям уровня $\Phi(x) = c = \text{const}$, его направление совпадает с направлением максимального роста $\Phi(x)$ в заданной точке (рис. 9.17). В точке минимума функции $\operatorname{grad} \Phi = 0$.

Определим итерационный процесс:

$$x^{(k+1)} = x^{(k)} - h \operatorname{grad} \Phi\left(x^{(k)}\right), \tag{9.6.2}$$

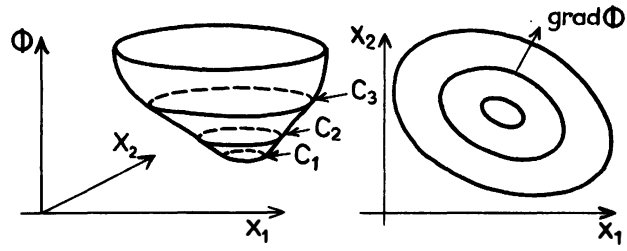

Рис. 9.17

где $h>0$ — шаг спуска, $x^{(0)}$ — заданное начальное приближение к точке минимума.

Заметим, что если $\operatorname{grad}\Phi(x^{(k)})\neq 0$, то для достаточно малых значений h

$$\ldots<\Phi(x^{(k)})<\ldots<\Phi(x^{(1)})<\Phi(x^{(0)}).$$

Формула (9.6.2) представляет собой метод градиентного спуска с постоянным шагом h спуска. Итерационный процесс (9.6.2) продолжается до выполнения какого-либо условия окончания алгоритма, например (9.6.1), либо следующего:

$$\|\operatorname{grad}\Phi(x^{(k+1)})\|<\varepsilon, \tag{9.6.3}$$

где ε — заданная точность.

Для примера рассмотрим поиск минимума

$$\Phi(x)=\frac{x_1^2}{4}+x_2^2 \tag{9.6.4}$$

методом градиентного спуска. Согласно (9.6.2), имеем

$$x_1^{(k+1)}=x_1^{(k)}-h\frac{x_1^{(k)}}{2},$$

$$x_2^{(k+1)}=x_2^{(k)}-h\,2x_2^{(k)}.$$

Пусть нулевое приближение $x^{(0)}=(x_1^{(0)}=1,\ x_2^{(0)}=1)$, шаг спуска $h=0{,}1$. Первые три шага спуска дают следующие приближения

$x_1^{(1)}=0{,}95;$ $x_1^{(2)}=0{,}9025;$ $x_1^{(3)}=0{,}8574;$
$x_2^{(1)}=0{,}80;$ $x_2^{(2)}=0{,}6400;$ $x_2^{(3)}=0{,}5120.$

За три шага $\Phi(x)$ уменьшилась с 1,25 до 0,446. Естественно поставить вопрос: нельзя ли увеличить шаг спуска так, чтобы быстрее спуститься к точке минимума (в примере — $x=(0,0)$)? Оказывается, что увеличивать h можно только до определенного предела. Слишком большой шаг может привести к увеличению $\Phi(x)$. В рассматриваемом примере $h=2$ приводит к $x^{(1)}=(x_1^{(1)}=0,\ x_2^{(1)}=-3)$ и значению $\Phi(x^{(1)})=9>\Phi(x^{(0)})$ (рис. 9.18). Таким образом, наиболее важным моментом в методе градиентного спуска оказывается выбор шага. Формула (9.6.2) с постоянным шагом h практически не применяется.

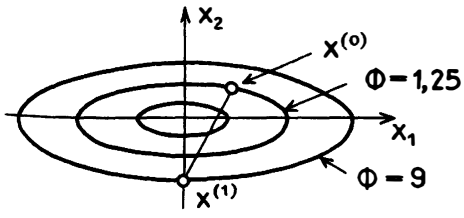

Рис. 9.18

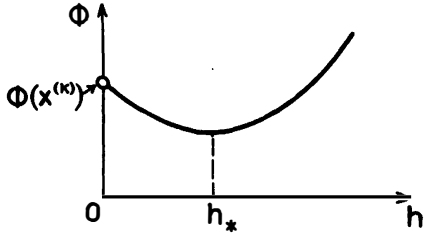

Рис. 9.19

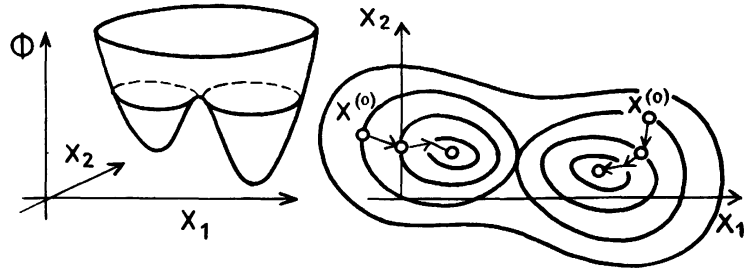

Рис. 9.20

Различные стратегии выбора шага $h=h(k)$ определяют разные варианты градиентного спуска.

Далее рассматривается вариант спуска, называемый наискорейшим градиентным спуском.

9.6.3. Метод наискорейшего градиентного спуска. Из (9.6.2) можно определить значение $\Phi(x)$ в точке $x^{(k+1)}$:

$$\Phi(x^{(k+1)})=\Phi(x^{(k)}-h\,\mathrm{grad}\,\Phi(x^{(k)})).$$

Если $x^{(k)}$ определено, то значение целевой функции Ф в следующей точке $x^{(k+1)}$ оказывается функцией только шага спуска (рис. 9.19).

Будем выбирать $h=h_*$ из условия, чтобы функция Ф за этот шаг максимально уменьшила свое значение:

$$\Phi(x^{(k+1)})=\min_h \Phi(x^{(k)}-h\,\mathrm{grad}\,\Phi(x^{(k)})).$$

Выбор шага h_* сводится к поиску минимума функции одной переменной.

В примере п. 9.6.2 выбор шага в точке $x^{(0)}$ сводится к следующей задаче:

$$\min_h\left[\frac{1}{4}\left(1-\frac{h}{2}\right)^2+(1-2h)^2\right],$$

откуда определяется шаг h наискорейшего спуска $h_*=10/9$.

Заметим, что методы градиентного спуска, вообще говоря, определяют локальный минимум целевой функции $\Phi(x)$. Это связано с зависимостью всего пути спуска $x^{(k)}$ от начального приближения $x^{(0)}$. Можно привести примеры, когда различные $x^{(0)}$ дают пути, приводящие к различным точкам локального минимума (рис. 9.20). Поэтому очень важно в задачах минимизации использовать всю имеющуюся информацию о зависимости целевой функции Ф от вектора x для правильного выбора начального приближения $x^{(0)}$.

9.6.4. Применение программы B5A1. Используем программу B5A1 для поиска минимума функции (9.6.4) по заданному начальному приближению $x^{(0)}=(1,1)$, спуск прекращается по условию

$$\| x^{(k+1)}-x^{(k)} \| \leqslant \varepsilon$$

в евклидовой норме. Программа может иметь следующий вид:

```
      REAL X(2),R,G(2),S,E,W(4)
      INTEGER N,I,K
      EXTERNAL F
      DATA X/1.,1./,N/2/,K/100/,E/1.E-4/
C     ОБРАЩЕНИЕ К ПРОГРАММЕ B5A1
      CALL B5A1(F,N,X,R,G,S,E,K,I,W)
C     ВЫВОД НА ТЕРМИНАЛ ВЕКТОРА X, ДОСТАВЛЯ-
C     ЮЩЕГО MIN F,
C     ЗНАЧЕНИЯ MIN F, ИНДЕКСА ОШИБОК I
      WRITE (5,) X,S,I
1     FORMAT (2X,2E13.6,'MINF=',E13.6,'I=',12)
      END
C     ПОДПРОГРАММА, ВЫЧИСЛЯЮЩАЯ F(X), ВЕКТОР
C     GRAD F(X)
      SUBROUTINE F(N,X,Y,G)
      REAL X(2),Y,G(2)
      INTEGER N
      Y=X(1)*X(1)/4.+X(2)*X(2)
      G(1)=X(1)/2.
      G(2)=2.*X(2)
      RETURN
      END
```

● 9.7. Метод золотого сечения

Выше, рассматривая покоординатный спуск и наискорейший градиентный спуск, мы отметили, что важным элементом этих методов является минимизация функций одной переменной. В настоящем разделе рассматривается метод деления интервала поиска минимума точками (метод золотого сечения), вычисления значения $\Phi(x)$ в них, сравнения значений и отбрасывания той части интервала, на которой заведомо отсутствует минимум. Такой подход аналогичен методу бисекции определения корня как по логической схеме, так и по минимальному требованию к гладкости минимизируемой функции $\Phi(x)$. Для метода золотого сечения даже не требуется непрерывности функции, могут существовать разрывы первого рода. Достаточно, чтобы $\Phi(x)$ была унимодальной.

9.7.1. Унимодальная функция. Функция $\Phi(x)$, заданная на интервале $a \leqslant x \leqslant b$, называется унимодальной на $[a, b]$, если существует единственная точка x_* минимума $\Phi(x)$

$$\Phi(x_*) = \min_{a \leqslant x \leqslant b} \Phi(x)$$

и если для любых двух точек $x_1, x_2 \in [a, b]$ выполняются соотношения:

из неравенств $x_1 < x_2 \leqslant x_*$ следует

$$\Phi(x_1) > \Phi(x_2),$$

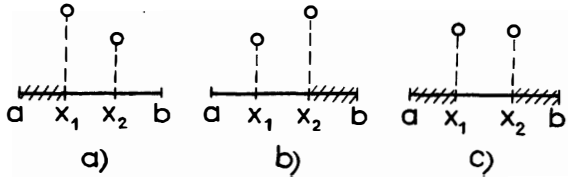

Рис. 9.21

из неравенств $x_2 > x_1 \geqslant x_*$ следует

$$\Phi(x_1) < \Phi(x_2).$$

Пусть известно, что $\Phi(x)$ унимодальна на $[a, b]$. Тогда по любым двум значениям $\Phi(x_1)$, $\Phi(x_2)$ можно указать интервал, в котором находится точка x_*, минимизирующая $\Phi(x)$, причем этот интервал имеет длину, меньшую первоначальной. Пусть для определенности $x_1 < x_2$, возможны следующие три варианта (рис. 9.21):

a) $\Phi(x_1) > \Phi(x_2)$;

b) $\Phi(x_1) < \Phi(x_2)$;

c) $\Phi(x_1) = \Phi(x_2)$.

В случае a) следует отбросить интервал $[a, x_1]$, в случае b)— $[x_2, b]$, в случае c)—интервалы $[a, x_1]$, $[x_2, b]$, поскольку на этих интервалах не может находиться x_*, в противном случае нарушается предположение об унимодальности $\Phi(x)$.

В зависимости от стратегии выбора двух точек x_1, x_2 на интервале имеются различные методы поиска минимума унимодальной функции, отличающиеся скоростью стягивания интервала неопределенности, содержащего x_*, к точке x_*.

9.7.2. Метод золотого сечения. Золотое сечение, открытое Евклидом, состоит в разбиении интервала $[a, b]$ точкой x_1 на две части таким образом, чтобы отношение длины всего интервала к большей части было равно отношению большей части к меньшей:

$$\frac{b-a}{b-x_1} = \frac{b-x_1}{x_1-a}.$$

Легко проверить, что золотое сечение производят две точки:

$$x_1 = a + (1-\tau)(b-a),$$
$$x_2 = a + \tau(b-a), \quad \tau = (1-\sqrt{5})/2.$$

Значение $\tau \simeq 0{,}618$.

Алгоритм метода золотого сечения следующий:

1) вычисляют значения x_1, x_2;

2) вычисляют $\Phi(x_1)$, $\Phi(x_2)$;

3) если $\Phi(x_1) \leqslant \Phi(x_2)$, то для дальнейшего деления оставляют интервал $[a, x_2]$;

4) если $\Phi(x_1) > \Phi(x_2)$, то для дальнейшего деления оставляют интервал $[x_1, b]$.

Процесс деления продолжают до тех пор, пока длина интервала неопределенности не станет меньше заданной точности ε.

Заметим, что точка x_1 производит золотое сечение интервала $[a, x_2]$, точка x_2—интервала $[x_1, b]$. Поэтому на оставшемся

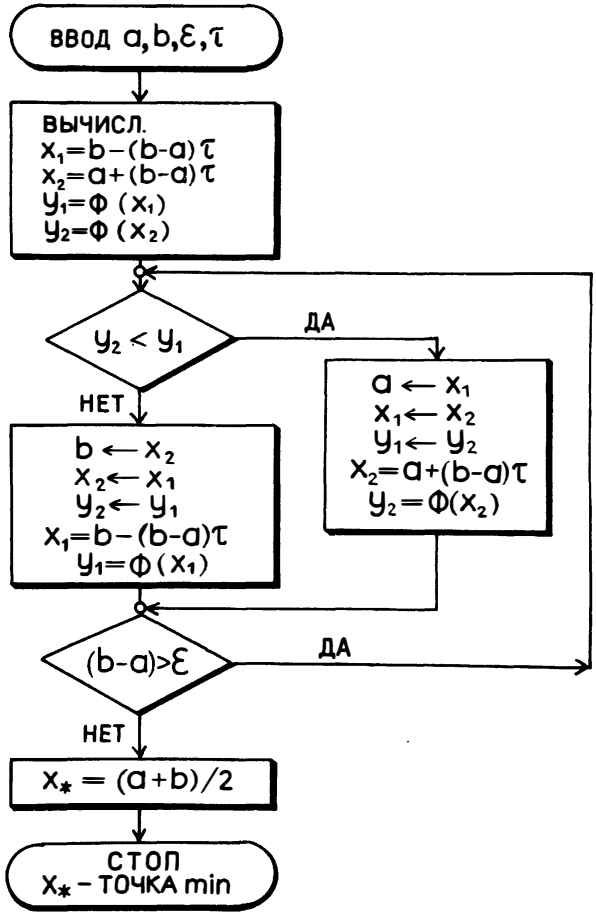

Рис. 9.22

интервале нужно определить одну точку, производящую золотое сечение. После этого процесс повторяется. На каждом шаге длина нового интервала неопределенности равна $\simeq 0{,}618$ длины старого интервала.

9.7.3. Схема алгоритма метода золотого сечения. Представим блок-схему алгоритма в следующем виде (рис. 9.22).

9.7.4. Применение программы B5A0. Для минимизации функции одной переменной можно использовать библиотечную программу B5A0. Пусть необходимо найти минимум функции

$$\Phi(x)=x^2+e^x$$

на интервале $[-1, 0]$ с относительной точностью определения x_*, равной 0,001, и абсолютной точностью 0,00001. Программа может иметь следующий вид:

```
      REAL E,E1,A,B,X,R
      INTEGER N,I
      EXTERNAL F
      DATA E,E1,A,B/1.E3,1.E5,1.,0./,N/100/
      DATA I/0/
C     ОБРАЩЕНИЕ К ПРОГРАММЕ B5A0
      CALL B5A0(F,E,E1,A,B,N,X,R,I)
C     ВЫВОД НА ТЕРМИНАЛ ТОЧКИ МИНИМУМА X И
C     ЗНАЧЕНИЯ F(X),
C     ИНДЕКСА ОШИБОК
      WRITE (5,1) X,R,I
1     FORMAT (2X,'X=',E13.6,'F(X)=',E13.6,'I=',12)
      END
C     ПОДПРОГРАММА, ВЫЧИСЛЯЮЩАЯ ЗНАЧЕНИЕ
C     ФУНКЦИИ
      SUBROUTINE F(X1,F1)
      REAL X1,F1
      F1=X1*X1+EXP(X1)
      RETURN
      END
```

Глава 10

ОБЫКНОВЕННЫЕ ДИФФЕРЕНЦИАЛЬНЫЕ УРАВНЕНИЯ

● 10.1. Дискретизация

10.1.1. Введение. В этой главе рассматриваются обыкновенные дифференциальные уравнения, наиболее общий вид которых — система дифференциальных уравнений

$$\frac{dy_i}{dx}=f_i(x,\ y_1,\ \ldots,\ y_n),\quad 1\leqslant i\leqslant n, \tag{10.1.1}$$

определенная на интервале $a\leqslant x\leqslant b$. Если дополнительные условия для однозначного нахождения решений $y_i(x)$ задаются в одной точке (для определенности $x=a$) интервала $[a,\ b]$, то это задача Коши, если в нескольких (двух краевых точках интервала $[a,\ b]$ $(x=a,\ x=b)$), то это краевая задача.

Далее задачи для дифференциальных уравнений (Коши, краевые) делятся на два класса: линейные и нелинейные.

Линейная задача определяется линейными дифференциальными уравнениями и линейными дополнительными условиями. Постановка нелинейной задачи содержит либо нелинейное уравнение, либо нелинейное условие, либо то и другое.

Всюду ниже предполагается, что решение поставленной задачи $y(x)$ для (10.1.1) существует и единственно.

В гл. 5 были описаны некоторые аналитические методы решения задач для (10.1.1), конечная цель которых — получение явной формулы точного решения $y(x)$. Заметим, что решение $y(x)$, вообще говоря, является элементом бесконечномерного пространства; для непрерывных по своим аргументам правых частей $f_i(x,\ y)$, например, $y(x)\in C^1\ [a,\ b]$. В реализации численных методов имеют дело только с элементами конечномерных пространств, поэтому необходимо перейти от систем уравнений (10.1.1) к системам алгебраических уравнений, в которых неизвестными являются векторы конечной размерности,— произвести дискретизацию исходной задачи.

Дискретизация линейных задач приводит к системам линейных алгебраических уравнений (см. гл. 8), нелинейных — к системам нелинейных алгебраических уравнений (см. гл. 9).

10.1.2. Этапы дискретизации. Для простоты изложения будем рассматривать одно уравнение

$$\frac{dy}{dx}=f(x,\ y) \tag{10.1.2}$$

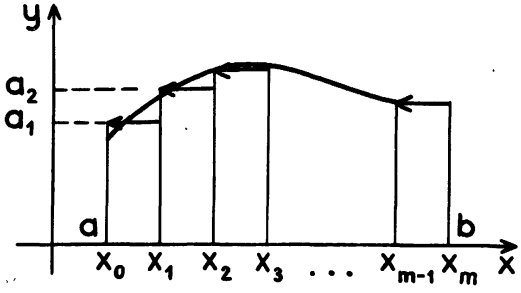

Рис. 10.1

с начальным условием

$$y(a)=y^{(0)}. \quad (10.1.3)$$

Пусть решение задачи (10.1.2), (10.1.3) $y(x)\in$ $\in Y=C^1[a,\ b]$.

Первый этап дискретизации состоит в переходе к конечномерному пространству функций Y_m, которое приближает любой элемент пространства решений $Y=C^1[a,\ b]$ тем точнее, чем больше размерность m конечномерного пространства.

Например, пространством Y_m может быть: 1) пространство полиномов $(m-1)$-й степени

$$y_m(x)=\{a_0+a_1x+\ldots+a_{m-1}x^{m-1}\}.$$

Выбирая коэффициенты a_i, можно по теореме Вейерштрасса при $m\to\infty$ сколь угодно точно приблизить любую непрерывную функцию $y(x)$;

2) пространство кусочно-постоянных функций

$$y_m(x)=\begin{cases} a_1,\ x_0<x\leqslant x_1,\ x_0=a,\\ a_2,\ x_1<x\leqslant x_2,\\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ a_m,\ x_{m-1}<x<x_m,\ x_m=b. \end{cases}$$

При $m\to\infty$ длина частичного интервала стремится к нулю и выбором a_i можно аппроксимировать непрерывную функцию $y(x)$ как угодно точно (рис. 10.1);

3) пространство параболических сплайнов (см. гл. 6)

$$y_M(x)=\begin{cases} S_i(x)=a_0^{(i)}+a_1^{(i)}x+a_2^{(i)}x^2,\ x_i\leqslant x\leqslant x_{i+1},\ 0\leqslant i\leqslant m-1,\\ S_i'(x_i)=S_{i+1}'(x_i),\ S_i(x_i)=S_{i+1}(x_i),\ 1\leqslant i\leqslant m-1. \end{cases}$$

Размерность пространства M определяется числом свободных параметров, при фиксированном m оно равно $M=3m-2(m-1)=$ $=m+2$.

Второй этап состоит в замене исходной задачи (10.1.2), (10.1.3) для функции $y(x)$ другой задачей, «близкой» для функции $y(x)$, но уже поставленной для $y_m(x)$. Фактически это задача нахождения чисел a_i, определяющих элемент Y_m. Различные численные методы решения дифференциальных уравнений отличаются именно этими двумя этапами дискретизации.

Третий этап — решение полученной системы линейных или нелинейных алгебраических уравнений, т. е. определение чисел a_i или, что то же самое, элемента $y_m(x)$.

Четвертый этап — восстановление по элементу $y_m(x)$ «близкой» функции $\bar{y}(x)$, принадлежащей Y. Функцию $\bar{y}(x)$ называют *приближенным решением исходной задачи.*

Пятый этап — оценка погрешности приближенного решения $\bar{y}$ в норме Y, т. е. оценка нормы

$$\| \bar{y}(x) - y(x) \|_Y.$$

Четвертый и пятый этапы часто заменяют следующими: определение проекции решения $y(x)$ на Y_m — определение $y(x) \in Y_m$. Затем производится оценка погрешности

$$\| y_m(x) - \tilde{y}(x) \|_{Y_m}$$

в норме Y_m.

10.1.3 Примеры дискретизации. 1) Рассмотрим задачу (10.1.2), (10.1.3) в предположении, что

$$a = 0,\ y^{(0)} = 0,$$

$$f(x, y) = f_0(x) + f_1(x) y,$$

где функции $f_i(x)$ являются полиномом не выше k-й степени;

$$f_i(x) = f_{i,0} + f_{i,1} x + \ \dots\ + f_{i,k} x^k, \quad i = 0,\ 1.$$

Проведем дискретизацию с помощью пространства полиномов, т. е.

$$y_{m+1}(x) = a_0 + a_1 x + \ \dots\ + a_m x^m. \tag{10.1.4}$$

Для того чтобы получить систему уравнений, определяющую a_i, подставим (10.1.4) в (10.1.2), (10.1.3) и приравняем в левой и правой частях равенств коэффициенты при одинаковых степенях x. Из (10.1.3) определяем

$$a_0 = 0. \tag{10.1.5}$$

Из (10.1.2) находим

$$a_1 + 2a_2 x + 3a_3 x^2 + \ \dots\ + m a_m x^{m-1} = f_{0,0} + f_{0,1} x + \ \dots\ + f_{0,k} x^k +$$
$$+ (f_{1,0} + f_{1,1} x + \ \dots\ + f_{1,k} x^k)(a_1 x + a_2 x^2 + \ \dots\ + a_m x^m).$$

Отсюда получаем соотношения для коэффициентов a_i полинома:

$$\begin{aligned}
a_1 &= f_{0,0}, \\
a_2 &= \frac{1}{2}(f_{0,1} + f_{1,0} a_1), \\
a_3 &= \frac{1}{3}(f_{0,2} + f_{1,0} a_2 + f_{1,1} a_1), \\
&\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\
a_m &= \frac{1}{m}(f_{0,m-1} + f_{1,0} a_{m-1} + f_{1,1} a_{m-2} + \ \dots\ + f_{1,m-2} a_1).
\end{aligned} \tag{10.1.6}$$

Уравнения (10.1.5), (10.1.6) и есть дискретная задача — система уравнений для a_i. Из вида этой системы следует, что для любого

m числа a_i могут быть последовательно определены. Причем если $m-2>k$, то общая формула для $a_m (m>k+2)$ следующая:

$$a_m=\frac{1}{m}\left(f_{1,0}\,a_{m-1}+f_{1,1}\,a_{m-2}+\ \dots\ +f_{1,k}\,a_{m-k-1}\right).$$

Пусть для примера

$$f_0(x)=1-x,\quad f_1(x)=1+x.$$

Тогда соотношения (10.1.5), (10.1.6) принимают вид

$$a_0=0,\ a_1=1,\ a_2=1,\ a_3=\frac{1}{3}(1+1)=\frac{2}{3},$$

$$a_4=\frac{1}{4}\left(1+\frac{2}{3}\right)=\frac{5}{12},\ a_5=\frac{1}{5}\left(\frac{2}{3}+\frac{5}{12}\right)=\frac{13}{60},\ \dots$$

$$a_m=\frac{1}{m}(a_{m-1}+a_{m-2}).$$

Из последнего равенства находим оценку

$$a_m\leqslant\frac{2}{m}a_{m-2},\quad m=3,\ 4,\ \dots,$$

Откуда получаем

$$a_{2k}\leqslant\frac{2^k}{1\cdot2\cdot4\ \dots\ (2k)}=\frac{1}{k!};\ k=1,\ 2,\ \dots,$$

$$a_{2k+1}\leqslant\frac{2^k}{1\cdot3\cdot5\ \dots\ (2k+1)},\quad k=1,\ 2,\ \dots.$$

Эти оценки показывают, что последовательность $y_m(x)$ при $m\to\infty$ равномерно сходится для любого b в интервале $0\leqslant x\leqslant b$ к точному решению исходной задачи Коши.

Четвертый этап дискретизации в данной задаче тривиален, можно положить

$$\bar{y}(x)=y_m(x).$$

Оценка погрешности $\bar{y}(x)$ выполняется следующим образом. Вводим функцию

$$\varepsilon(x)=y(x)-y_m(x).$$

Для функции $\varepsilon(x)$ получаем сходящийся ряд

$$\varepsilon(x)=\sum_{k=m+1}^{\infty}a_k x^k,$$

где a_k определяются формулами (10.1.6). Абсолютная погрешность в равномерной норме $C[0,\ b]$ равна

$$\varepsilon=\max_{0\leqslant x\leqslant b}|\varepsilon(x)|=\sum_{k=m+1}^{\infty}a_k b^k\leqslant\sum_{k=m+1}^{\infty}\frac{b^k}{k!}.$$

2) Рассмотрим ту же задачу с $a=0$, $y^{(0)}=1$, но в качестве пространства Y_m возьмем пространство кусочно-постоянных фун-

кций. Поскольку в точках разрыва у функций $y_m(x)$ не существует производной, перейдем от задачи (10.1.2), (10.1.3) к эквивалентному интегральному уравнению

$$y(x)=1+\int_0^x [f_0(s)+f_1(s)y(s)]\,ds. \tag{10.1.7}$$

Подставляя функцию $y_m(x)$ в (10.1.7), получаем систему соотношений

$$a_1=1+\int_0^x [f_0(x)+f_1(s)a_1]\,ds, \quad 0<x\leqslant x_1,$$

$$a_2=a_1+\int_{x_1}^x [f_0(s)+f_1(s)a_2]\,ds, \quad x_1<x\leqslant x_2,$$

. . . . . .

$$a_m=a_{m-1}+\int_{x_{m-1}}^x [f_0(s)+f_1(s)a_m]\,ds, \quad x_{m-1}<x\leqslant b.$$

Эти соотношения противоречивы, поскольку слева в равенствах — константы, а справа — функции. Если $|x_i-x_{i-1}|$ достаточно малы, то можно эту систему соотношений заменить «близкой», но такой, что полученная система однозначно будет определять a_i. Положим

$$a_1=1+\int_0^{x_1} [f_0(s)+f_1(s)a_1]\,ds,$$

$$a_2=a_1+\int_{x_1}^{x_2} [f_0(s)+f_1(s)a_2]\,ds,$$

. . . . . .

$$a_m=a_{m-1}+\int_{x_{m-1}}^{x_m} [f_0(s)+f_1(s)a_m]\,ds.$$

Отсюда определяются последовательно все a_i. Действительно,

$$a_1=\frac{1+\int_0^{x_1} f(s)\,ds}{1-\int_0^{x_1} f_1(s)\,ds},$$

$$a_2=\frac{a_1+\int_{x_1}^{x_2} f_0(s)\,ds}{1-\int_{x_1}^{x_2} f_1(s)\,ds},$$

. . . . . .

$$a_m=\frac{a_{m-1}+\int_{x_{m-1}}^{x_m} f_0(s)\,ds}{1-\int_{x_{m-1}}^{x_m} f_1(s)\,ds}.$$

Интегралы здесь можно вычислить с помощью квадратурных формул. Положим $f_0(x) \equiv 0$, $h = x_i - x_{i-1}$, $f_1(x) = \lambda = \text{const}$. Тогда при условии, что $|\lambda h| < 1$, получаем явную формулу

$$a_i = \frac{1}{(1-\lambda h)^i}, \quad 1 \leqslant i \leqslant m.$$

Заметим, что $h = x_m/m$, отсюда

$$a_m = \left(1 - \lambda \frac{x_m}{m}\right)^{-m} = \left(1 - \lambda \frac{b}{m}\right)^{-m}. \tag{10.1.8}$$

Точное решение исходной задачи в данном случае

$$y(x) = e^{\lambda x}.$$

Заметим, что при $m \to \infty$ в точке $x = b$ из (10.1.8) следует существование предела, равного

$$e^{\lambda b} = \lim_{m \to \infty} \left(1 - \frac{\lambda b}{m}\right)^{-m},$$

совпадающего со значением точного решения в точке $x = b$.

Определим кусочно-постоянную функцию $y(x)$ по точному решению

$$\tilde{y}(x) = e^{\lambda x_i}, \; x_{i-1} < x \leqslant x_i, \; 1 \leqslant i \leqslant m.$$

Погрешность $Y_m(x)$ в равномерной норме равна

$$\max_{1 \leqslant k \leqslant m} \left| \left(1 - \lambda \frac{x_k}{k}\right)^{-k} - e^{\lambda x_k} \right| = \left| \left(1 - \lambda \frac{b}{m}\right)^{-m} - e^{\lambda b} \right|.$$

10.1.4. Дискретизация по формулам численного дифференцирования. Дискретизация упрощается, если ограничиться поиском решений дифференциальных уравнений в конечном числе точек интервала $[a, b]$.

1) Разобьем интервал $[a, b]$ точками x_i с постоянным шагом $h = x_i - x_{i-1}$ $(1 \leqslant i \leqslant m)$, $x_0 = a$, $x_m = b$.

2) Точное решение $y(x)$ дифференциального уравнения

$$\frac{dy}{dx} = f(x, y)$$

в точках x_i принимает значения

$$y(x_i), \quad 1 \leqslant i \leqslant m,$$

а дифференциальное уравнение приводит к равенствам

$$\frac{dy}{dx}(x_i) = f(x_i, y(x_i)). \tag{10.1.9}$$

Если в формуле (10.1.9) значение производной

$$\frac{dy}{dx}(x_i), \quad 0 \leqslant i \leqslant m,$$

заменить на «близкое» выражение, которое определяется через $y(x_i)$, то получим систему m уравнений с m неизвестными, являющуюся дискретной задачей, которая соответствует исходной задаче.

Например, можно положить (см. ниже формулы численного дифференцирования)

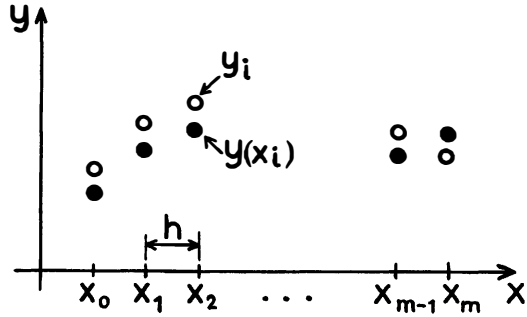

Рис. 10.2

$$\frac{dy}{dx}(x_i)=\frac{y(x_{i+1})-y(x_i)}{h}+O(h).$$

Тогда уравнение (10.1.9) принимает вид

$$y(x_{i+1})-y(x_i)=hf(x_i, y(x_i))+O(h^2).$$

Отбрасывая слагаемое $O(h^2)$, получаем «близкое» уравнение

$$y_{i+1}-y_i=hf(x_i, y_i). \qquad (10.1.10)$$

Здесь y_i — точное решение дискретной задачи (10.1.10). Пусть найден вектор y_i, $0\leqslant i\leqslant m$. Это и есть приближенные значения для величин $y(x_i)$ (рис. 10.2).

Множество точек x_i, $0\leqslant i\leqslant m$, называют сеткой, x_i — узлами сетки, h — шагом сетки. Уравнение (10.1.10) — это разностное уравнение или разностная схема.

Для оценки погрешности вводится норма вектора $y=(y_i, 0\leqslant i\leqslant m)$, которая называется сеточной нормой, например:

равномерная норма

$$\|y\|=\max_{0\leqslant i\leqslant m}|y_i|,$$

среднеквадратичная норма

$$\|y\|=\left(\sum_{i=1}^{m}hy_i^2\right)^{1/2}.$$

Ниже рассматривается только равномерная норма. Погрешность решения дискретной задачи задается величиной

$$\|y_i-y(x_i)\|.$$

10.1.5. Формулы численного дифференцирования. Пусть x — любой из узлов x_i интервала $[a, b]$. Тогда другие узлы могут быть вычислены по формулам

$$x+kh,\ \ k=0, \pm1, \pm2, \ldots \qquad (10.1.11)$$

Приближенное вычисление производной функции $y(x)$

$$\frac{d^sy}{dx^s}(x),\ \ s\geqslant1,$$

в узле x по значениям функции в узлах $y(x+kh)$ называется *численным дифференцированием.*

Предположим, что $y(x)\in C^2[a, b]$, $x<b$. Разложим $y(x+h)$ в ряд Тейлора до членов второго порядка. Имеем

$$y(x+h)=y(x)+\frac{dy}{dx}(x)\,h+O(h^2).$$

Отсюда находим, что

$$\frac{dy}{dx}(x)=\frac{y(x+h)-y(x)}{h}+O(h).$$

Учитывая, что x — это узел x_i, получаем двухточечную формулу для первой производной — дифференцирование вперед:

$$\frac{dy}{dx}(x_i)=\frac{y(x_{i+1})-y(x_i)}{h}+O(h),\quad 0\leqslant i\leqslant m-1. \tag{10.1.12}$$

Обычно под формулой численного дифференцирования понимают приближенное равенство

$$\frac{dy}{dx}(x_i)=\frac{y(x_{i+1})-y(x_i)}{h}.$$

Разность

$$R=\frac{dy}{dx}(x_i)\frac{y(x_{i+1})-y(x_i)}{h},$$

вычисленная на функции $y(x)\in C^2[a, b]$, называется *погрешностью аппроксимации.* Для (10.1.12) имеем

$$R=O(h),\quad h\to 0.$$

Точностью (или порядком точности) *аппроксимации* называется порядок степени h, с которым $R(h)$ стремится к нулю при $h\to 0$. Для (10.1.12) порядок точности — первый.

Аналогично выводится двухточечная формула для первой производной — дифференцирование назад:

$$\frac{dy}{dx}(x_i)=\frac{y(x_i)-y(x_{i-1})}{h}+O(h),\quad 1\leqslant i\leqslant m. \tag{10.1.13}$$

Более точной является двухточечная формула для первой производной — центральная разность

$$\frac{dy}{dx}(x_i)=\frac{y(x_{i+1})-y(x_{i-1})}{2h}+O(h^2),\quad 1\leqslant i\leqslant m-1. \tag{10.1.14}$$

Формула (10.1.14) верна, если $y(x)\in C^3[a, b]$.

Для первой производной можно выписать многоточечные формулы, например трехточечная формула имеет вид

$$\frac{dy}{dx}(x_i)=\frac{-y(x_{i-2})+4y(x_{i+1})-3y(x_i)}{2h}+O(h^2).$$

В общем виде многоточечная формула имеет (при достаточной гладкости $y(x)$) вид

$$\frac{dy}{dx}(x_i)=\sum_{k=0}^{m} a_k y(x_k)+O(h^p),$$

где коэффициенты a_k выбираются так, чтобы порядок точности формулы был p. Аналогичный подход применяется и для высших производных.

Теорема 10.1. *Пусть* $y(x)\in C^4[a, b]$. *Тогда для второй производной справедлива формула численного дифференцирования*

$$\frac{d^2y}{dx^2}(x_i)=\frac{y(x_{i+1})-2y(x_i)+y(x_{i-1})}{h^2}+O(h^2) \qquad (10.1.15)$$

для внутренних узлов $1\leqslant i\leqslant m-1$.

Доказательство. Разложим $y(x+h)$, $y(x-h)$ в ряд Тейлора до членов четвертого порядка

$$y(x+h)=y(x)+\frac{dy}{dx}(x)h+\frac{1}{2}\frac{d^2y}{dx^2}(x)h^2+\frac{1}{6}\frac{d^3y}{dx^3}(x)h^3+O(h^4),$$

$$y(x-h)=y(x)-\frac{dy}{dx}(x)h+\frac{1}{2}\frac{d^2y}{dx^2}(x)h^2-\frac{1}{6}\frac{d^3y}{dx^3}(x)h^3+O(h^4).$$

Подставляя правые части в выражение

$$\frac{y(x+h)-2y(x)+y(x-h)}{h^2},$$

получаем формулу (10.1.15), что и требовалось доказать.

10.1.6. Простейшие разностные схемы задачи Коши. Аппроксимация, устойчивость, сходимость. Рассмотрим исходную задачу Коши на сетке x_i, $0\leqslant i\leqslant m$:

$$\frac{dy}{dx}(x_i)=f(x_i, y(x_i)),\quad y(x_0)=y^{(0)}.$$

Заменяя левую часть дифференциального уравнения формулами (10.1.12)—(10.1.14), получаем:

$$\frac{y(x_{i+1})-y(x_i)}{h}=f(x_i, y(x_i))+O(h),\quad 0\leqslant i\leqslant m-1,$$

$$\frac{y(x_i)-y(x_{i-1})}{h}=f(x_i, y(x_i))+O(h),\quad 1\leqslant i\leqslant m,$$

$$\frac{y(x_{i+1})-y(x_{i-1})}{2h}=f(x_i, y(x_i))+O(h^2),\quad 1\leqslant i\leqslant m-1.$$

Если отбросить погрешность аппроксимации производных в этих соотношениях, то получим разностные схемы:

$$\frac{y_{i+1}-y_i}{h}=f(x_i, y_i),\quad 0\leqslant i\leqslant m-1,\quad y_0=y^{(0)}; \qquad (10.1.16)$$

$$\frac{y_i - y_{i-1}}{h} = f(x_i, y_i), \quad 1 \leqslant i \leqslant m, \quad y_0 = y^{(0)}; \tag{10.1.17}$$

$$\frac{y_{i+1} - y_{i-1}}{2h} = f(x_i, y_i), \quad 1 \leqslant i \leqslant m-1, \quad y_0 = y^{(0)}. \tag{10.1.18}$$

Разностные схемы (10.1.16), (10.1.17) аппроксимируют исходную задачу Коши для достаточно гладких функций на сетке с первым порядком точности, схема (10.1.18) — со вторым порядком точности.

Схема (10.1.16) называется *явной разностной схемой*, схемы (10.1.17), (10.1.18) — *неявными*. Термины связаны с тем, что формула (10.1.16) дает возможность вычислить значение y_{i+1} в следующей точке сети x_{i+1} по явной формуле

$$y_{i+1} = y_i + hf(x_i, y_i),$$

если известно значение y_i, в то время как формулы (10.1.17), (10.1.18) задают значения в следующей точке как неявную функцию от значений в предыдущих точках.

Схема (10.1.18) — незамкнутая разностная схема в том смысле, что система уравнений (10.1.18) имеет $m+1$ неизвестное y_i, $0 \leqslant i \leqslant m$, и только m уравнений. Замкнуть систему можно по-разному. Чтобы не потерять второй порядок точности аппроксимации, обычно заменяют

$$f(x_i, y_i) = \frac{1}{2}\left[f(x_{i-1}, y_{i-1}) + f(x_{i+1}, y_{i+1})\right];$$

шаг сетки увеличивают вдвое и получают следующую неявную схему:

$$\frac{y_{i+1} - y_i}{h} = \frac{1}{2}\left[f(x_i, y_i) + f(x_{i+1}, y_{i+1})\right], \tag{10.1.19}$$

$$1 \leqslant i \leqslant m, \quad y_0 = y^{(0)}.$$

В качестве примера построения разностных схем рассмотрим модельную задачу

$$\frac{dy}{dx} = \lambda y, \quad y(0) = y^{(0)}, \quad 0 \leqslant x \leqslant b, \tag{10.1.20}$$

где λ — константа. Точное решение (10.1.20):

$$y(x) = y^{(0)} e^{\lambda x}.$$

Разностная схема (10.1.16) примет вид

$$y_{i+1} = y_i + h\lambda y_i, \quad 0 \leqslant i \leqslant m-1, \quad y_0 = y^{(0)}. \tag{10.1.21}$$

Разностная схема (10.1.17)

$$y_i = y_{i-1} + h\lambda y_i, \quad 1 \leqslant i \leqslant m, \quad y_0 = y^{(0)}. \tag{10.1.22}$$

Разностная схема (10.1.19)

$$y_{i+1} = y_i + \frac{h\lambda}{2}\left[y_i + y_{i+1}\right], \quad 1 \leqslant i \leqslant m, \quad y_0 = y^{(0)}. \tag{10.1.23}$$

Точные решения уравнений (10.1.23)—(10.1.23) следующие:

$$y_{i+1}=(1+h\lambda)\,y_i,\quad y_i=(1+h\lambda)^i y^{(0)},\quad 0\leqslant i\leqslant m;$$

$$y_i=\frac{1}{1-h\lambda}\,y_{i-1},\quad y_i=\frac{1}{(1-h\lambda)^i}\,y^{(0)},\quad 0\leqslant i\leqslant m,$$

при условии $h\lambda\neq 1$;

$$y_{i+1}=\frac{1+h\lambda/2}{1-h\lambda/2}\,y_i,\quad y_i=\left(\frac{1+h\lambda/2}{1-h\lambda/2}\right)^i y^{(0)},\quad 0\leqslant i\leqslant m,$$

при условии $h\lambda\neq 2$. Точное решение (10.1.20) в узлах сетки принимает значения

$$y(x_i)=e^{\lambda x_i}y^{(0)}=e^{h\lambda i}y^{(0)},\quad 0\leqslant i\leqslant m.$$

Отсюда находим погрешность решения соответствующей разностной схемы:

$$r=\left((1+h\lambda)^i-e^{h\lambda i}\right)y^{(0)},$$

$$r=\left((1+h\lambda)^{-i}-e^{h\lambda i}\right)y^{(0)},\quad h\lambda\neq 1,$$

$$r=\left(\left(\frac{1+h\lambda/2}{1-h\lambda/2}\right)^i-e^{h\lambda i}\right)y^{(0)},\quad h\lambda\neq 2.$$

Обозначим $P(h\lambda)$ коэффициент преобразования y_i в y_{i+1} при переходе из узла сетки x_i в следующий узел. Для трех рассматриваемых схем имеем:

$$P(h\lambda)=(1-h\lambda),$$

$$P(h\lambda)=(1-h\lambda)^{-1},$$

$$P(h\lambda)=(1+h\lambda/2)(1-h\lambda/2)^{-1}.$$

Тогда погрешность решения разностных схем запишется в форме

$$r=y^{(0)}\left(P-e^{h\lambda}\right)\left(P^{i-1}+P^{i-2}e^{h\lambda}+\ldots+Pe^{h\lambda(i-2)}+e^{h\lambda(i-1)}\right),$$

$$1\leqslant i\leqslant m. \qquad (10.1.24)$$

В этом выражении первый сомножитель связан с начальным условием, второй — с погрешностью аппроксимации дифференциального уравнения разностным на одном шаге, третий — с накоплением погрешности по шагам $i=1, 2, \ldots, m$.

Разностная схема называется *устойчивой на модельной задаче* (10.1.20), если накопление погрешности по шагам не зависит от числа шагов, т. е.

$$\left|\sum_{k=1}^{i} P^{i-k}e^{h\lambda(k-1)}\right|\leqslant c,\quad 1\leqslant i\leqslant m, \qquad (10.1.25)$$

где константа c не зависит от m.

Разностная схема называется *сходящейся на модельной задаче* (10.1.20), если

$$\max_{1\leqslant i\leqslant m}|y_i-y(x_i)|\to 0 \text{ при } h\to 0,$$

сходящейся с p-м порядком, если

$$\max_{1\leqslant i\leqslant m} |y_i - y(x_i)| = O(h^p), \quad h\to 0.$$

Теорема 10.2. *Если разностная схема устойчива на модельной задаче* (10.1.20) *и имеет p-й порядок точности аппроксимации дифференциального уравнения на шаге, то схема сходится с p-м порядком к точному решению.*

Доказательство. По условию,

$$P - e^{h\lambda} = O(h^p).$$

Но тогда, переходя в (10.1.24) к оценкам модуля r и учитывая (10.1.25), получаем

$$\max_{1\leqslant i\leqslant m} |r| \leqslant |y^{(0)}|\,|P - e^{h\lambda}|\,c = O(h^p),$$

что и требовалось доказать.

Коротко утверждение теоремы формулируется так: аппроксимация и устойчивость схемы дают ее сходимость.

Пусть в модельной задаче $\lambda < 0$; пусть, кроме того,

$$|P(h\lambda)| \leqslant q < 1. \tag{10.1.26}$$

Тогда схема устойчива. Действительно, в этом случае ($1 \leqslant i \leqslant m$) имеем

$$|P^{i-1} + P^{i-2}e^{h\lambda} + \ldots + e^{h\lambda(i-1)}| \leqslant |P^{i-1} + P^{i-2} + \ldots + 1| \leqslant \frac{1}{1-q} = c,$$

т. е. выполняется неравенство (10.1.25). Для рассматриваемых трех схем условие устойчивости (10.1.26) — ограничение на выбор шага h — принимает вид

$$|1 + h\lambda| \leqslant q < 1; \quad \left|\frac{1}{1-h\lambda}\right| \leqslant q < 1; \quad \left|\frac{1+h\lambda/2}{1-h\lambda/2}\right| \leqslant q < 1.$$

Заметим, что при больших значениях λ эти условия ограничивают шаг h только для первой разностной схемы.

Пусть в модельной задаче $\lambda > 0$. Тогда все рассматриваемые схемы неустойчивы. Действительно, в этом случае для достаточно малых h имеем $P > 1$, далее

$$|P^{i-1} + P^{i-2}e^{h\lambda} + \ldots + e^{h\lambda(i-1)}| > |P^{i-1} + P^{i-2} + \ldots + 1| = \frac{P^{i-1} - 1}{P - 1}, \quad 2 \leqslant i \leqslant m.$$

Но правая часть этого неравенства стремится к бесконечности при $i = m \to \infty$.

Не следует считать, что неустойчивые схемы нельзя применять для вычислений. Важным является не свойство устойчивости, а свойство сходимости разностной схемы. Поэтому если в (10.1.25) $c(h) \to \infty$, $h \to 0$, но при этом

$$(P(h\lambda) - e^{h\lambda})\,c(h) \to 0, \quad h \to 0,$$

то схема является сходящейся.

В качестве примера рассмотрим схему (10.1.21) при $y^{(0)}=1$, $b=1$ для двух уравнений:

$$\frac{dy}{dx}=-y,$$

здесь $\lambda=-1$, схема устойчива;

$$\frac{dy}{dx}=y,$$

здесь $\lambda=+1$, схема неустойчива.

Для устойчивой схемы при $h=0{,}1$; $0{,}01$ разность между точным решением в точке $x=1$ и разностным соответственно равны:

$$e^{-1}-(1-0{,}1)^{10}=0{,}019 \text{ при } h=0{,}1;$$
$$e^{-1}-(1-0{,}01)^{100}=0{,}0018 \text{ при } h=0{,}01.$$

Для неустойчивой (но сходящейся) схемы

$$e-(1+0{,}1)^{10}=0{,}12 \text{ при } h=0{,}1,$$
$$e-(1+0{,}01)^{100}=0{,}013 \text{ при } h=0{,}01.$$

Из приведенных числовых значений легко заметить, что скорость сходимости устойчивой схемы на порядок по h выше, чем сходящейся, но неустойчивой схемы.

● 10.2. Задача Коши. Методы Рунге — Кутта

В **10.1** приведены разностные схемы, которые имеют невысокий порядок точности аппроксимации по шагу h. Такие схемы на практике используются редко, поскольку они обладают медленной сходимостью к точному решению. Однако при построении более точных разностных схем необходимо, чтобы правые части дифференциальных уравнений были достаточно гладкими. Обычно число непрерывных производных правой части уравнения должно быть на единицу больше порядка точности. Поэтому недостаточная гладкость правых частей уравнений может быть главной причиной использования разностных схем невысокого порядка точности.

10.2.1. Применение рядов Тейлора в построении разностных схем. Покажем, как с помощью рядов Тейлора можно построить схему произвольного порядка точности аппроксимации дифференциального уравнения на шаге.

Рассмотрим задачу Коши

$$\frac{dy}{dx}=f(x, y),\quad y(a)=y^{(0)} \tag{10.2.1}$$

на интервале $[a, b]$. Разобьем интервал на m отрезков узлами x_i с постоянным шагом h. Точное решение задачи (10.2.1) в точке x_{i+1} на любом из частичных интервалов $[x_i, x_{i+1}]$, $0\leqslant i\leqslant m-1$, можно представить в виде ряда Тейлора с центром в точке x_i.

Имеем для функций $f(x, y)$ ($p+1$ раз непрерывно дифференцируемых по обоим аргументам) формулу

$$y(x_{i+1})=y(x_i)+y'(x_i)\,h+\frac{y''(x_i)}{2!}h^2+\ldots+\frac{y^{(p)}(x_i)}{p!}h^p+O(h^{p+1}). \qquad (10.2.2)$$

Производные в (10.2.2) вычисляются согласно дифференциальному уравнению (10.2.1) следующим образом:

$$y'(x)=f(x, y),$$

$$y''(x)=\frac{\partial f}{\partial x}(x, y)+\frac{\partial f}{\partial y}(x, y)\,y'=\frac{\partial f}{\partial x}(x, y)+\frac{\partial f}{\partial y}(x, y)f,$$

$$y'''(x)=\frac{\partial}{\partial x}\left(\frac{\partial f}{\partial x}+\frac{\partial f}{\partial y}f\right)+\frac{\partial}{\partial y}\left(\frac{\partial f}{\partial x}+\frac{\partial f}{\partial y}f\right)f,$$

.....

$$y^{(p)}(x)=\frac{\partial}{\partial x}(y^{(p-1)})+\frac{\partial}{\partial y}(y^{(p-1)})f. \qquad (10.2.3)$$

Здесь следует положить $y=y(x)$ — точное решение (10.2.1), а затем подставить $x=x_i$.

Если отбросить в (10.2.2) остаточный член, то получим дискретное уравнение

$$y_{i+1}=y_i+hf(x_i y_i)+\frac{h^2}{2}\left[\frac{\partial f}{\partial x}(x_i y_i)+\frac{\partial f}{\partial y}(x_i y_i), f(x_i y_i)\right]+\ldots+$$

$$+\frac{h^p}{p!}\left[\frac{\partial^{p-1}}{\partial x^{p-1}}f(x_i y_i)+\ldots+\frac{\partial^{p-1}}{\partial y^{p-1}}(x_i y_i)f^{p-1}\right], \qquad (10.2.4)$$

где $0\leqslant i\leqslant m-1$, $y_0=y^{(0)}$ задано в (10.2.1). Уравнение (10.2.4) определяет двухточечную явную разностную схему. По формуле (10.2.4) можно последовательно определить все значения y_i (рис. 10.3), начиная с y_1 и кончая y_m.

Погрешность аппроксимации (10.2.1) разностным уравнением (10.2.4), очевидно, равна величине отброшенного остаточного члена, т. е. $O(h^{p+1})$ при $h\to 0$. Если взять $p=1$, то получаем схему

$$y_{i+1}=y_i+hf(x_i, y_i), \qquad (10.2.5)$$

совпадающую с (10.1.16). Это *явный метод Эйлера*. Если взять $p=2$, то получаем схему

$$y_{i+1}=y_i+h\left[f(x_i y_i)+\frac{h}{2}(f'_x(x_i y_i)+f'_y(x_i y_i)f(x_i y_i))\right], \qquad (10.2.6)$$

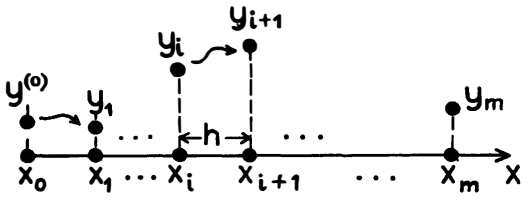

Рис. 10.3

порядок аппроксимации которой $O(h^3)$. Это так называемый *исправленный метод Эйлера*.

Для получения разностных схем методом рядов Тейлора необходи-

мо иметь аналитические выражения полных производных по x от функции $f(x, y)$, а именно (10.2.3). Это может быть достаточно громоздкой задачей. В таком случае ряды Тейлора не применяются, а используются методы Рунге — Кутта.

10.2.2. Методы Рунге — Кутта. Идея методов Рунге — Кутта состоит в том, чтобы представить дискретную задачу, соответствующую (10.2.1), в виде

$$y_{i+1}=y_i+h\varphi(x_i, y_i, h),\quad 0\leqslant i\leqslant m-1, \tag{10.2.7}$$

где функция $\varphi(x, y, h)$ приближала бы отрезок ряда Тейлора (10.2.2) с точностью $O(h^{p+1})$ и в то же время не содержала бы производных $f(x, y)$. Таким образом, в основе методов Рунге — Кутта лежит подгонка ряда Тейлора.

Заметим, что метод Рунге — Кутта первого порядка ($p=1$) — это метод Эйлера, так как здесь в вычислениях используются только значения $f(x, y)$.

Покажем на примере метода Рунге — Кутта второго порядка, как выводятся формулы, т. е. определяется функция $\varphi(x, y, h)$ в (10.2.6). Будем искать φ в следующем виде:

$$\varphi(x, y, h)=c_1f(x, y)+c_2f(x+ha_2, y+hb_{21}f(x, y)). \tag{10.2.8}$$

В формуле (10.2.8) c_1, c_2, a_2, b_{21} — пока неизвестные константы, которые находятся из сравнения с рядом Тейлора. Разложим второе слагаемое в (10.2.8) в ряд Тейлора с центром в точке (x_i, y_i) до членов порядка $O(h^2)$; получаем

$$\varphi(x_i, y_i, h)=c_1f(x_i, y_i)+c_2f(x_i, y_i)+c_2\frac{\partial f}{\partial x}(x_i, y_i)\,ha_2+$$

$$+c_2\frac{\partial f}{\partial y}(x_i, y_i)\,hb_{21}f(x_i, y_i)+O(h^2). \tag{10.2.9}$$

Соответствующий отрезок ряда Тейлора (10.2.2), (10.2.3) имеет вид (10.2.6). Отсюда, подставляя (10.2.9) в (10.2.7) и приравнивая коэффициенты при соответствующих слагаемых в (10.2.6), получаем систему равенств

$$\begin{aligned} &c_1+c_2=1,\\ &c_2a_2=1/2,\\ &c_2b_{21}=1/2. \end{aligned}$$

Эта система трех уравнений с четырьмя неизвестными имеет решение $c_1=1-\alpha$, $c_2=\alpha$, $a_2=b_{21}=1/2\alpha$, зависящее от произвольного параметра $\alpha\neq 0$. Обычно применяется значение $\alpha=1/2$. Тогда метод Рунге — Кутта второго порядка можно записать в следующей форме:

$$\begin{cases} y_{i+1}=y_i+\dfrac{h}{2}(k_1+k_2),\quad 0\leqslant i\leqslant m-1,\\ k_1=f(x_i, y_i),\\ k_2=f(x_i+h, y_i+hk_1), \end{cases} \tag{10.2.10}$$

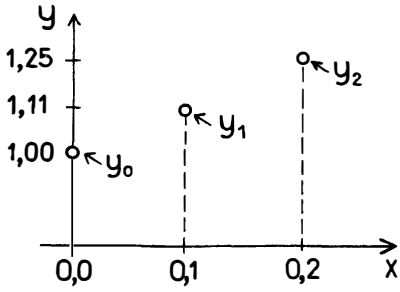

Рис. 10.4

где $y_0=y^{(0)}$. Заметим, что метод второго порядка имеет погрешность аппроксимации $O(h^3)$, для перехода из точки x_i в точку x_{i+1} требует дважды вычислить правую часть дифференциального уравнения $f(x, y)$, чтобы определить k_1, k_2.

Например, пусть имеем задачу Коши

$$\frac{dy}{dx}=x^2+y^2,\ y(0)=1,\ 0\leqslant x\leqslant 0{,}2. \qquad (10.2.11)$$

Выполняя два шага в соответствии с (10.2.10) при $h=0{,}1$, получаем (рис. 10.4):

$$y_0=1,$$

$$\begin{cases} y_1=1+0{,}05(k_1+k_2)=\underline{1{,}111}, \\ k_1=1, \\ k_2=0{,}1^2+(1+0{,}1)^2=1{,}22; \end{cases}$$

$$\begin{cases} y_2=1{,}111+0{,}05(k_1+k_2)=\underline{1{,}2515307}, \\ k_1=(0{,}1)^2+(1{,}111)^2=1{,}244321, \\ k_2=(0{,}2)^2+(1{,}111+0{,}1244321)^2=1{,}5662924. \end{cases}$$

Подчеркнуты предполагаемые верные значащие цифры точного решения $y(0{,}1)$, $y(0{,}2)$.

Для метода Рунге — Кутта произвольного порядка p по аналогии с (10.2.7), (10.2.8) формула записывается в форме

$$\begin{cases} y_{i+1}=y_i+h\sum\limits_{j=1}^{p} c_j k_j,\ 0\leqslant i\leqslant m-1, \\ k_1=f(x_i, y_i), \\ k_j=f\left(x_i+ha_j,\ y_i+h\sum\limits_{s=1}^{j-1} b_{js}k_s\right),\ 2\leqslant j\leqslant m. \end{cases}$$

Для конкретного значения p соответствующие константы c_j, a_j, b_{js} находятся по той же схеме, которая использовалась при $p=2$. Популярным среди методов Рунге — Кутта является метод четвертого порядка. Метод имеет погрешность аппроксимации $O(h^5)$, соответствующие формулы имеют вид

$$
\begin{aligned}
&y_{i+1}=y_i+\frac{h}{6}(k_1+2k_2+2k_3+k_4),\ 0\leqslant 1\leqslant m-1,\\
&k_1=f(x_i,\ y_i),\\
&k_2=f\left(x_i+\frac{h}{2},\ y_i+\frac{h}{2}k_1\right),\\
&k_3=f\left(x_i+\frac{h}{2},\ y_i+\frac{h}{2}k_2\right),\\
&k_4=f(x_i+h,\ y_i+hk_3).
\end{aligned}
\tag{10.2.12}
$$

На каждом шаге необходимо четыре раза вычислить правую часть дифференциального уравнения. Выполним один шаг для задачи (10.2.11). Имеем

$$
\begin{cases}
y_0=1,\\
y_1=1+\dfrac{0{,}1}{6}(k_1+2k_2+2k_3+k_4)=\underline{1{,}1114}629,\\
k_1=1,\\
k_2=(0{,}05)^2+(1+0{,}05)^2=1{,}105,\\
k_3=(0{,}05)^2+(1+0{,}05\cdot 1{,}105)^2=1{,}1160525,\\
k_4=(0{,}1^2)+(1+0{,}1\cdot 1{,}116)^2=1{,}2456663.
\end{cases}
$$

Подчеркнуты предполагаемые верные значащие цифры точного решения $y(0{,}1)$.

Если рассматривается задача Коши для системы дифференциальных уравнений

$$
\begin{aligned}
&\frac{dy_i}{dx}=f_i(x,\ y_1,\ ...,\ y_n),\ 1\leqslant i\leqslant n,\\
&y_i(a)=y_i^{(0)},
\end{aligned}
$$

то формулы Рунге — Кутта четвертого порядка аналогичны скалярному случаю. Отличие в записи следующее: индекс i указывает теперь номер компоненты вектора, а индекс j — номер узла x_j, $0\leqslant j\leqslant m-1$, в котором соответствующий вектор $y^{(j)}$ имеет компоненты $y^{(j)}=(y_1^{(j)},\ y_2^{(j)},\ ...,\ y_n^{(j)})$, векторы k имеют также n компонент, например $k_1=(k_{1,1},\ k_{1,2},\ ...,\ k_{1,n})$. Формулы Рунге — Кутта четвертого порядка для системы дифференциальных уравнений следующие:

$$
\begin{aligned}
&y^{(j+1)}=y^{(j)}+\frac{h}{6}(k_1+2k_2+2k_3+k_4),\ 0\leqslant j\leqslant m-1,\\
&k_{1,i}=f_i(x_j,\ y_1^{(j)},\ ...,\ y_n^{(j)}),\ 1\leqslant i\leqslant n,\\
&k_{2,i}=f_i\left(x_j+\frac{h}{2},\ y_1^{(j)}+\frac{h}{2}k_{1,1},\ ...,\ y_n^{(j)}+\frac{h}{2}k_{1,n}\right),\\
&k_{3,i}=f_i\left(x_j+\frac{h}{2},\ y_1^{(j)}k_{2,1},\ ...,\ y_n^{(j)}+\frac{h}{2}k_{2,n}\right),\\
&k_{4,i}=f_i\left(x_j+h,\ y_1^{(j)}+hk_{3,1},\ ...,\ y_n^{(j)}+\frac{h}{2}k_{3,n}\right).
\end{aligned}
$$

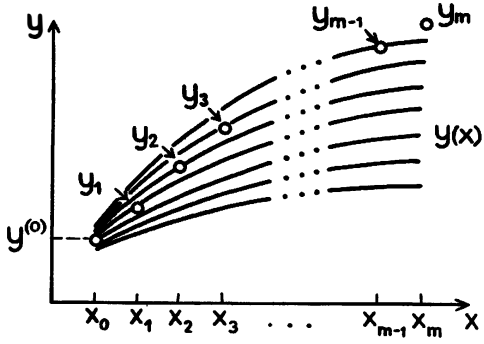

Рис. 10.5

10.2.3. Оценка погрешности. Погрешность аппроксимации дифференциального уравнения разностным для метода Рунге—Кутта p-го порядка есть $O(h^{p+1})$, поэтому погрешность метода на одном шаге (локальная погрешность) дается оценкой

$$|y_{i+1}-y(x_{i+1})|\leqslant O(h^{p+1}),$$

если в точке x_i значение точного решения совпадает с y_i, т. е.

$$y_i=y(x_i), \tag{10.2.13}$$

при условии, что правая часть дифференциального уравнения $f(x, y)$ $p+1$ раз непрерывно дифференцируема в окрестности решения $y(x)$. Интервал $[a, b]$, на котором ищется приближение к решению, содержит $m=(b-a)/h$ шагов. Но только на первом шаге выполняется равенство (10.2.13). На следующем шаге равенство можно сохранить, причем под $y(x_i)$ следует понимать точное решение, но не с условием $y(x_0)=y^{(0)}$, а с условием $y(x_1)=y_1$ и т. д. (рис. 10.5). Погрешность на всем интервале $[a, b]$ (глобальная погрешность) определяется формулой

$$\max_{1\leqslant i\leqslant m}|y_i-y(x_i)|.$$

Чтобы оценить эту величину, необходимо уметь выражать разность решений дифференциальных уравнений через их разность в начальной точке x_i.

Теорема 10.3. *Пусть $Y_1(x)$ и $Y_2(x)$—решения дифференциального уравнения*

$$\frac{dy}{dx}=f(x, y).$$

Пусть также функция $f(x, y)$ непрерывно дифференцируема. Тогда

$$Y_2(x_{i+1})-Y_1(x_{i+1})=$$
$$=(Y_2(x_i)-Y_1(x_i))\exp\left(\int_{x_i}^{x_{i+1}} f'_y(x, \bar{y}(x))\,dx\right), \tag{10.2.14}$$

где $\bar{y}(x)$—функция, расположенная между Y_1 и Y_2.

Доказательство. Поскольку Y_1, Y_2—решения, имеем

$$\frac{dY_1}{dx}=f(x, Y_1);\ \frac{dY_2}{dx}=f(x, Y_2).$$

Вычитая из второго равенства первое и применяя теорему Лагранжа, находим

$$\frac{d}{dx}(Y_2-Y_1)=f(x,\ Y_2)-f(x,\ Y_1)=f'_y(x,\ \bar{y}(x))(Y_2-Y_1),$$

где $\bar{y}(x)$ расположена между $Y_1(x)$ и $Y_2(x)$. Решая полученное линейное дифференциальное уравнение относительно $z(x)=Y_2-Y_1$ с начальным условием $Y_2(x_i)-Y_1(x_1)$, получаем

$$z(x)=z(x_i)\exp\left(\int\limits_{x_i}^{x} f'_y(x,\ \bar{y}(x))\,dx\right).$$

Из последнего равенства при $x=x_{i+1}$ имеем (10.2.14), что и требовалось доказать.

Обозначим решение дифференциального уравнения с условием $Y(x_0)=y^{(0)}$ через $Y_0(x)$; с условием $Y(x_1)=y_1$ — через $Y_1(x)$ и т. д., с условием $Y(x_{m-1})=y_{m-1}$ — через $Y_{m-1}(x)$. Тогда погрешность в любом узле x_k можно представить в виде

$$R_k=y_k-y(x_k)=\sum_{i=1}^{k}\left(Y_i(x_i)-Y_{i-1}(x_i)\exp\left(\int\limits_{x_i}^{x_k} f'_y(x,\ \bar{y}(x))\,dx\right)\right). \qquad (10.2.15)$$

Теорема 10.4. *Пусть погрешность на шаге h метода Рунге—Кутта имеет оценку*

$$|y_{i+1}-y(x_{i+1})|\leqslant ch^{p+1}$$

с условием (10.2.13); *пусть также*

$$f'_y(x,\ y)\leqslant L(x),\quad L(x)\geqslant 0, \qquad (10.2.16)$$

в некоторой окрестности решения $y(x)$. Тогда для достаточно малых h глобальная оценка погрешности следующая:

$$\max_{1\leqslant i\leqslant m}|y_i-y(x_i)|\leqslant\left[(b-a)\,ch^p\right]\exp\left(\int\limits_a^b L(x)\,dx\right). \qquad (10.2.17)$$

Доказательство. Заметим, что

$$\max_{1\leqslant i\leqslant m}|y_i-y(x_i)|=\max_{1\leqslant k\leqslant m}|R_k|,$$

где R_k определяется (10.2.15). Но в силу (10.2.16)

$$|R_k|\leqslant\sum_{i=1}^{k}|Y_i(x_i)-Y_{i-1}(x_i)|\exp\int\limits_{x_i}^{x_k} f_y(x,\ \bar{y}(x))\,dx\leqslant$$

$$\leqslant\sum_{i=1}^{k}(ch^{p+1})\exp\int\limits_{x_i}^{x_k}L(x)\,dx\leqslant\sum_{i=1}^{m}(ch^{p+1})\exp\left(\int\limits_a^b L(x)\,dx\right)=$$

$$=cmh^{p+1}\exp\left(\int\limits_a^b L(x)\,dx\right).$$

Учитывая, что $m=(b-a)/h$, получаем

$$\max_{1\leqslant k\leqslant m}|R_k|\leqslant c(b-a)\,h^p\exp\left(\int\limits_a^b L(x)\,dx\right),$$

что и требовалось доказать.

Неравенство (10.2.17) дает оценку погрешности методов Рунге — Кутта порядка p и указывает, что при $h \to 0$ решения разностных схем методов сходятся к точному решению.

Однако если $L(x) > 0$ и интервал интегрирования $[a, b]$ велик, то коэффициент при h^p в (10.2.17) может быть столь большой, что для достижения заданной точности придется брать очень малый шаг h. При этом вычислительная погрешность (например, округления) может превосходить погрешность метода Рунге — Кутта.

Заметим, что если в (10.2.16) $L(x) \leqslant \lambda < 0$, т. е.

$$f'_y(x, y) \leqslant \lambda < 0, \quad a \leqslant x \leqslant b,$$

то оценка $|R_k|$ в доказательстве теоремы примет вид

$$|R_k| \leqslant c(b-a) h^p \exp(\lambda(b-a)),$$

а так как функция $xe^{\lambda x}$ на всей полуоси $0 \leqslant x \leqslant \infty$ ограничена $xe^{\lambda x} < \dfrac{1}{|\lambda|} e^{-1}$, то оценка погрешности

$$\max_{1 \leqslant k \leqslant m} |R_k| \leqslant \frac{ce^{-1}}{|\lambda|} h^p$$

оказывается не зависящей от длины интервала интегрирования.

10.2.4. Оценка погрешности по правилу Рунге. Правило Рунге уже использовалось при оценке погрешности численного интегрирования (см. п. 7.4.2). Обобщим это правило на интегрирование дифференциальных уравнений. Заметим, что численное интегрирование можно рассматривать как интегрирование дифференциального уравнения в том частном случае, когда

$$f(x, y) = f(x), \quad \frac{dy}{dx} = f(x), \quad y(a) = 0$$

для определения значения $y(b) = \int_a^b f(x)\, dx$.

Если не удается получить для функции $f(x, y)$ значения c, $L(x)$, необходимые, чтобы применить оценку погрешности (10.2.17), то применяют правило Рунге.

Для оценки погрешности сравнивают приближенные решения, полученные при различных шагах сетки. При этом используется следующее предположение: глобальная погрешность метода порядка p в точке x_i представляется в виде

$$y_i - y(x_i) = w(x_i) h^p + O(h^{p+1}); \tag{10.2.18}$$

здесь $x_i = a + ih$. Если провести вычисления тем же методом порядка p, но с шагом, вдвое меньшим, то, согласно (10.2.18), получим

$$y_{2i} - y(x_{2i}) = w(x_{2i}) \frac{h^p}{2^p} + O(h^{p+1}); \tag{10.2.19}$$

здесь $x_{2i}=a+2i\frac{h}{2}=x_i$. Эти два соотношения позволяют найти, вычитая (10.2.19) из (10.2.18) значение $w(x_{2i})$,

$$y_i-y_{2i}=w(x_{2i})\left[h^p-\frac{h^p}{2^p}\right]+O(h^{p+1}),$$

$$w(x_{2i})=\frac{2^p(y_i-y_{2i})}{h^p(2^p-1)}+O(h).$$

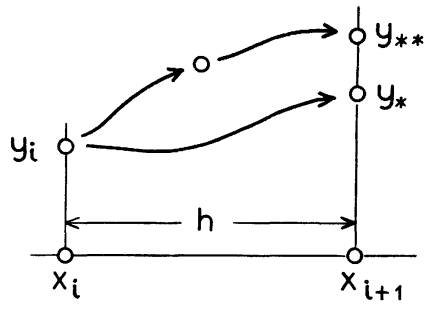

Рис. 10.6

Подставляя последнее выражение в (10.2.19), получаем

$$y_{2i}-y(x_{2i})=\frac{y_i-y_{2i}}{2^p-1}+O(h^{p+1}).$$

Таким образом, с точностью до $O(h^{p+1})$ при $h\to 0$ погрешность метода имеет вид

$$y_{2i}-y(x_{2i})\cong\frac{y_i-y_{2i}}{2^p-1},$$

где y_i — приближенное значение, полученное в точке x_{2i} с шагом h; y_{2i} — с шагом $h/2$; p — порядок метода. Например, для метода Рунге — Кутта второго порядка

$$|y_{2i}-y(x_{2i})|\simeq\frac{1}{3}|y_i-y_{2i}|;$$

для четвертого порядка

$$|y_{2i}-y(x_{2i})|\simeq\frac{1}{15}|y_i-y_{2i}|.$$

Правило Рунге лежит в основе алгоритмов интегрирования дифференциальных уравнений с автоматическим выбором шага по заданной локальной точности. На каждом шаге при переходе от x_i к x_{i+1} вычисления производятся дважды: с шагом h и шагом $h/2$ (рис. 10.6). Полученные значения y_* и y_{**} служат для сравнения достигнутой точности на этом шаге

$$\delta=\frac{|y_*-y_{**}|}{2^p-1}$$

с заданной ε; в случае систем уравнений модуль заменяется нормой $\|y_*-y_{**}\|$. Если величина δ больше ε, то шаг h уменьшается вдвое и процедура повторяется; если δ значительно меньше ε (указывается предел), то шаг h увеличивается вдвое, если δ меньше ε, но незначительно, то переходим в точку x_{i+2} с тем же шагом h.

Автоматический выбор шага для специальных систем дифференциальных уравнений сопровождают более совершенной стратегией контроля точности (не на каждом шаге), чем описанная выше [20].

10.2.5. Применение программы B6A0. Пусть необходимо проинтегрировать систему уравнений

$$\frac{dy_1}{dx}=\sin(x+y_1y_2),$$

$$\frac{dy_2}{dx}=\cos(x^2-y_1+y_2)$$

на интервале $0\leqslant x\leqslant 1$ с заданной абсолютной точностью $\varepsilon=10^{-4}$ и начальными условиями $y_1(0)=2$, $y_2(0)=(1)$. Вывод на терминал значений $y_1(x)$, $y_2(x)$ необходимо производить с шагом $H=0{,}1$. Программа может иметь следующий вид:

```
      REAL X,B,Y(2),E,W(2,7),R(2)
      INTEGER N,J,I
      EXTERNAL F,O
      DATA X,B/0.,1./,Y/2.,1./,E/1.E-4/
      DATA N,J,I/2,1,0/
C     ОБРАЩЕНИЕ К ПРОГРАММЕ B6A0
      CALL B6A0(X,B,N,Y,E,J,F,O,W,I)
      END
C     ВНЕШНЯЯ ПОДПРОГРАММА, ВЫЧИСЛЯЮЩАЯ ПРА-
C     ВЫЕ ЧАСТИ УРАВНЕНИЙ
      SUBROUTINE F(X,Y,R)
      REAL X,Y(2),R(2)
      R(1)=SIN(X+Y(1)*Y(2))
      R(2)=COS(X*X-Y(1)+Y(2))
      RETURN
      END
C     ВНЕШНЯЯ ПОДПРОГРАММА, ОСУЩЕСТВЛЯЮЩАЯ
C     ВЫВОД ЗНАЧЕНИЙ Y(X)
      SUBROUTINE O(X,Y)
      REAL X,Y(2),H
      DATA H/0.1/
      WRITE (5,1) X,Y(1),Y(2)
1     FORMAT (2X,'X=',F4.1,'Y(1)=',E13.6,'Y(2)=',E13.6)
      X=X+H
      RETURN
      END
```

● 10.3. Жесткие уравнения

10.3.1. Примеры жестких уравнений. Понятие жесткого уравнения связано с жестким условием устойчивости разностных схем для таких уравнений. Рассмотрим модельную задачу Коши на интервале $0\leqslant x\leqslant 1$:

$$\frac{dy}{dx}=\lambda y+a,\quad y(0)=y^{(0)}.$$

Пусть λ — достаточно большое отрицательное число. Разобьем интервал $[0,1]$ постоянным шагом h и запишем схему Эйлера (10.1.21):

$$y_{i+1}=y_1+h\lambda y_i+ah,\quad 0\leqslant i\leqslant m-1,\quad h=1/m.$$

Для нее условие устойчивости (10.1.26) принимает вид

$$|1+h\lambda|\leqslant q<1.$$

При больших значениях $|\lambda|$ это условие ограничивает выбор шага h величинами порядка

$$h\sim\frac{1}{|\lambda|}. \tag{10.3.1}$$

Условие (10.3.1) — жесткое условие на шаг h, отсюда и возник термин «жесткие» уравнения.

При этом уравнения с большими $\lambda>0$ не относятся к классу жестких по той причине, что условие типа (10.3.1) практически не играет роли. Выбор шага обусловлен требованием хорошего приближения к точному решению. Действительно, задача

$$\frac{dy}{dx}=100y+100,\quad y(0)=0$$

имеет точное решение $y=\exp(100x)-1$. Для того чтобы приблизить $y(1)$ с точностью до двух верных значащих цифр, в схеме следует взять $h=10^{-6}$, тогда получим

$$y_{10^6}=(1+h\cdot 100)^{1/h}-1=2{,}67\cdot 10^{43},$$
$$y(1)=2{,}69\cdot 10^{43},$$

в то время как условие типа (10.3.1) рекомендует выбирать шаг $h\sim 10^{-2}$.

Совершенно по-другому обстоит дело в случае больших отрицательных λ. Задача

$$\frac{dy}{dx}=-100y+100,\quad y(0)=0$$

имеет точное решение $y=1-\exp(-100x)$ (рис. 10.7), из вида которого следует, что при значениях $x\gtrsim 0{,}05$ точное решение отличается от $y=1$ в третьем знаке. Решение разностной схемы

$$y_i=1-(1-h\cdot 100)^i$$

описывает приближенно точное решение только если $|1-h\cdot 100|<1$, откуда следует, что должно быть выполнено жесткое условие $h<0{,}02$. Если оно нарушено, например $h=0{,}05$, то будем иметь следующие значения y_i: $y_0=0$, $y_1=5$, $y_2=15$, ..., ничего общего не имеющие с точным решением.

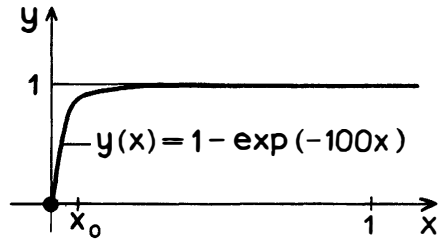

Рис. 10.7

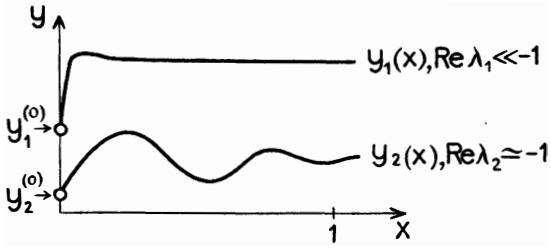

Рис. 10.8

Сравнивая точное и приближенное решения, отмечаем, что в приближенном решении слагаемое $(1-h\cdot 100)^i$, которое аппроксимирует $\exp(-100x)$, не позволяет увеличить шаг, хотя для значений $x>0{,}05$ вклад этих слагаемых в решение оценивается третьим знаком.

Этот факт демонстрирует общую ситуацию, характерную для жестких уравнений: решение содержит слагаемое, вклад которого мал почти на всем интервале интегрирования, но для сохранения устойчивости методы, не ориентируемые на эти уравнения, требуют аппроксимировать это слагаемое достаточно точно, выбирая малый шаг h. Для точности представления решения, как будет видно из дальнейшего, такой маленький шаг не нужен.

Более содержательный пример жестких уравнений дает система линейных уравнений

$$\frac{dy}{dx}=Ay+f,\quad 0\leqslant x\leqslant 1,\quad y(0)=y^{(0)},$$

где y, f — векторы, A — постоянная матрица, у которой все собственные значения λ_i имеют отрицательные вещественные части такие, что

$$\mathrm{Re}\,\lambda_i\leqslant\lambda_0<0,\quad 1\leqslant i\leqslant n,$$

$$\min_{1\leqslant i\leqslant n}|\mathrm{Re}\,\lambda_i|\simeq 1,\quad \max_{1\leqslant i\leqslant n}|\mathrm{Re}\,\lambda_i|\gg 1,\quad \max|\mathrm{Im}\,\lambda_i|\simeq 1. \qquad (10.3.2)$$

В этом случае решение $y(x)$ содержит быстропеременные слагаемые вида $\exp(\lambda_i x)$, у которых $\mathrm{Re}\,\lambda_i\ll -1$, и медленно изменяющиеся слагаемые, у которых $\mathrm{Re}\,\lambda_i\simeq -1$.

Здесь возникает та же ситуация, что и в первом примере: условие устойчивости в явной схеме Эйлера не дает возможности интегрировать с шагом h, определяемым точностью медленно изменяющихся слагаемых для тех значений x, где $\exp(\lambda_i x)$ с $\mathrm{Re}\,\lambda_i\ll 1$ мало отличаются от нуля (рис. 10.8).

В технических задачах роль независимого переменного обычно играет время. Тогда быстро протекающие процессы (затухающие во времени) на фоне медленно протекающих порождают явление, описываемое жесткими уравнениями.

Для общих систем нелинейных дифференциальных уравнений понятие жестких уравнений вводится по аналогии с приведенными выше примерами. Пусть имеем задачу

$$\frac{dy_i}{dx}=f_i(x, y_1, \ldots, y_n),\quad 0\leqslant x\leqslant 1,$$

$$y_i(0)=y_i^{(0)},\quad 1\leqslant i\leqslant n,$$

точное решение которой $y(x)=(y_1(x), ..., y_n(x))$. Вычислим якобиан:

$$A(x)=\left(\frac{\partial f_i}{\partial y_j}(x, y_1(x), ..., y_n(x)\right), \quad 1\leqslant i, j\leqslant n.$$

Если матрица $A(x)$ при некоторых $x\in[0,1]$ обладает свойствами (10.3.2), то исходная система уравнений относится к классу жестких.

Для характеристики уравнений относительно явления жесткости вводится коэффициент жесткости

$$S(x)=\frac{\max \operatorname{Re}(-\lambda_i(x))}{\min \operatorname{Re}(-\lambda_i(x))}, \quad 0\leqslant x\leqslant 1,$$

где $\lambda_i(x)$ — собственные значения матрицы $A(x)$. На практике система считается жесткой, если $S(x)>10$, однако в задачах химической кинетики, управления, электрических цепей коэффициенты $S(x)$ достигают величин $\sim 10^6$ и более.

В качестве примера рассмотрим известную модельную задачу химической кинетики:

$$\begin{aligned}
&\frac{dy_1}{dx}=-0{,}04y_1+10^4y_2y_3, && y_1(0)=1,\\
&\frac{dy_2}{dx}=0{,}04y_1-10^4y_2y_3-3\cdot 10^7y_2^2, && y_2(0)=0,\\
&\frac{dy_3}{dx}=3\cdot 10^7y_2^2, && y_3(0)=0.
\end{aligned} \tag{10.3.3}$$

Эта система имеет интеграл, равный

$$y_1+y_2+y_3=1,$$

поэтому сводится к системе второго порядка, для которой коэффициент жесткости $S(x)\sim 10^4$ на интервале $0\leqslant x\leqslant 1$.

Не случайно в этом разделе интервал интегрирования строго фиксирован единичным $0\leqslant x\leqslant 1$. Большая длина интервала $a\leqslant x\leqslant b$ интегрирования может привести к тому, что система уравнений должна рассматриваться как жесткая, хотя $S(x)$ и не велик на $[a, b]$.

Это легко понять из простого примера. Пусть имеем задачу

$$\frac{dy}{dx}=-y, \quad y(0)=y^{(0)}, \quad 0\leqslant x\leqslant b.$$

Коэффициент жесткости $S(x)=1$ на всем интервале $[0, b]$. Но если привести заменой переменной $x_1=x/b$ эту задачу к нормированному интервалу $0\leqslant x_1\leqslant 1$, то получим

$$\frac{dy}{dx_1}=-by, \quad y(0)=y^{(0)}, \quad 0\leqslant x_1\leqslant 1,$$

при $b\gg 1$ жесткое уравнение.

Отсюда целесообразно сделать следующий практический вывод: интегрирование задачи Коши на длинных интервалах может

привести к явлению жесткости, которое следует учитывать при выборе метода решения.

10.3.2. Неявные методы решения жестких уравнений. Покажем на примере, как неявная разностная схема (10.1.22) позволяет интегрировать жесткое уравнение с шагом h, выбираемым по заданной точности, а не из соображений устойчивости. Рассмотрим задачу

$$\frac{dy}{dx}=-100y+100,\quad y(0)=0,\quad 0\leqslant x\leqslant 1,$$

для которой схема (10.1.22) следующая:

$$y_i=y_{i-1}+h(-100y_i+100),\quad 1\leqslant i\leqslant m,\quad h=1/m.$$

Эта схема безусловно устойчива при $\lambda<0$, так как выполняется неравенство (10.1.26). Решение разностного уравнения

$$y_i=1-\frac{1}{(1+100h)^i},\quad 0\leqslant i\leqslant m,$$

указывает, что точное решение $y(x)=1-\exp(-100x)$ будет аппроксимировано с желаемой точностью без ограничений на величину шага h. Например, при $h=0{,}2$ имеем:

$$y_0=0,\quad y_1=0{,}952,\quad y_2=0{,}997,\quad y_3=0{,}999,$$
$$y(1)=0,\quad y(0{,}2)=0{,}999,\quad y(0{,}4)=0{,}999,\quad y(0{,}6)=0{,}999;$$

наихудшую относительную точность представления решения на первом шаге (порядка 5%).

Общий подход к решению жестких уравнений состоит в использовании неявных методов (неявных разностных схем).

Аналогом схемы (10.1.22) для нелинейного уравнения

$$\frac{dy}{dx}=f(x,\ y)$$

является неявная схема Эйлера

$$y_i=y_{i-1}+f(x_i,\ y_i),\quad 1\leqslant i\leqslant m, \tag{10.3.4}$$

которая на каждом шаге требует решения нелинейного уравнения (10.3.4) относительно y_i, что представляет, вообще говоря, не простую задачу. Эта задача еще более усложняется для системы нелинейных дифференциальных уравнений (см. гл. 9). Естественно, решение нелинейных уравнений связано с большими затратами машинного времени на каждом шаге, чем в явных методах. Но за счет возможности значительно увеличить шаг h общий объем вычислений в случае использования неявных методов может быть существенно меньше, чем явных.

10.3.3. Применение программы B6A1. Для иллюстрации применения программы B6A1 рассмотрим задачу Коши для жестких уравнений (10.3.3) на интервале $0\leqslant x\leqslant 1$. Зададим относительную

точность $\varepsilon = 10^{-3}$. Будем выводить на терминал значения $y_1(x)$, $y_2(x)$, $y_3(x)$ с шагом $H = 0{,}05$. Программа может иметь следующий вид:

```
      REAL X,B,Y(3),E,W(3,21),R(3),Z(3,3)
      INTEGER N,J,I,M,K
      EXTERNAL F,O,P
      DATA X,B/0.,1./,Y/1.,0.,0./,E/1.E-3/
      DATA N,J,I,M,K/3,0,0,1,21/
C     ОБРАЩЕНИЕ К ПРОГРАММЕ B6A1
      CALL B6A1(X,B,N,Y,E,J,F,M,P,O,W,K,I)
      END
C     ВНЕШНЯЯ ПОДПРОГРАММА, ВЫЧИСЛЯЮЩАЯ
C     ПРАВЫЕ ЧАСТИ УРАВНЕНИЙ
      SUBROUTINE F(X,Y,R)
      REAL X,Y(3),R(3),A,B,C
      DATA A,B,C/0.04,1.E4,3.E7/
      R(1)=-A*Y(1)+B*Y(2)*Y(3)
      R(2)=A*Y(1)-B*Y(2)*Y(3)-C*Y(2)*Y(2)
      R(3)=C*Y(2)*Y(2)
      RETURN
      END
C     ВНЕШНЯЯ ПОДПРОГРАММА, ОСУЩЕСТВЛЯЮЩАЯ
C     ВЫВОД ЗНАЧЕНИЙ Y(X)
      SUBROUTINE O(X,Y)
      REAL X,Y(3),H
      DATA H/0.05/
      WRITE (5,1) X,Y(1),Y(2),Y(3)
1     FORMAT (2X,'X=',F5.2,3E13.6)
      X=X+H
      RETURN
      END
C     ВНЕШНЯЯ ПОДПРОГРАММА, ВЫЧИСЛЯЮЩАЯ
C     ЯКОБИАН
      SUBROUTINE P(X,Y,Z)
      REAL X,Y(3),Z,(3,3),A,B,C
      DATA A,B,C/0.04,1.E4,3.E7/
      Z(1,1)=-A
      Z(1,2)=B*Y(3)
      Z(1,3)=B*Y(2)
      Z(2,1)=A
      Z(2,2)=-B*Y(3)-C*Y(2)*2.
      Z(2,3)=-B*Y(2)
      Z(3,1)=0.
      Z(3,2)=C*Y(2)*2.
      Z(3,3)=0.
      RETURN
      END
```

● 10.4. Краевые задачи

10.4.1. Линейное дифференциальное уравнение второго порядка. Разностная схема. Рассмотрим линейное дифференциальное уравнение второго порядка

$$a(x)\frac{d^2y}{dx^2}+b(x)\frac{dy}{dx}+c(x)y=f(x) \tag{10.4.1}$$

на интервале $a\leqslant x\leqslant b$ с условиями на краях интервала

$$\begin{aligned}&\alpha_0 y(a)+\alpha_1\frac{dy}{dx}(a)=\alpha_2,\\ &\beta_0 y(b)+\beta_1\frac{dy}{dx}(b)=\beta_2.\end{aligned} \tag{10.4.2}$$

Будем предполагать, что функции $a(x)$, $b(x)$, $c(x)$, $f(x)$ достаточно гладкие на $[a, b]$, $a(x)\neq 0$ на $[a, b]$, константы α_i, β_i таковы, что $\alpha_0^2+\alpha_1^2\neq 0$, $\beta_0^2+\beta_1^2\neq 0$.

Прежде чем заниматься численным решением исходной краевой задачи, целесообразно привести дифференциальное уравнение к простейшей форме. Для этого разделим (10.4.1) на $a(x)$. Получим

$$\frac{d^2y}{dx^2}+\frac{b(x)}{a(x)}\frac{dy}{dx}+\frac{c(x)}{a(x)}y=\frac{f(x)}{a(x)}. \tag{10.4.3}$$

Введем новую искомую функцию $z(x)$ с помощью замены

$$y=\exp\left(-\frac{1}{2}\int_a^x\frac{b(x)}{a(x)}dx\right)z. \tag{10.4.4}$$

Тогда для $z(x)$ из (10.4.3) имеем уравнение

$$\frac{d^2z}{dx^2}+q(x)z=r(x), \tag{10.4.5}$$

где

$$q(x)=\frac{c(x)}{a(x)}-\frac{1}{4}\left(\frac{b(x)}{a(x)}\right)^2-\frac{1}{2}\frac{d}{dx}\left(\frac{b(x)}{a(x)}\right),$$

$$z(x)=\frac{f(x)}{a(x)}\exp\left(\frac{1}{2}\int_a^x\frac{b(x)}{a(x)}d(x)\right),$$

и краевые условия

$$\begin{aligned}&\alpha_0 z(a)+\alpha_1\frac{dz}{dx}(a)=\alpha_2,\\ &\beta_0 z(b)+\beta_1\frac{dz}{dx}(b)=\beta_2\end{aligned} \tag{10.4.6}$$

со значениями α_i, β_i, отличными от тех, которые заданы в (10.4.2).

Если интеграл

$$I(x)=\int_a^x \frac{b(x)}{a(x)}dx$$

не берется аналитически, то тогда функция $I(x)$ может быть определена с помощью численного интегрирования (см. гл. 7). Мы же будем предполагать, что известна формула для $I(x)$, а следовательно, краевая задача, которую необходимо решать,— это уравнение (10.4.5) с условиями (10.4.6).

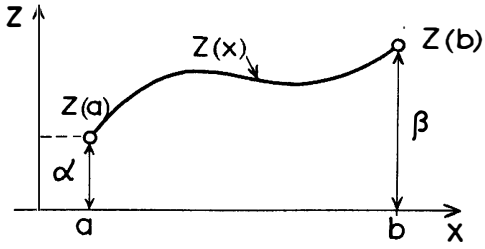

Рис. 10.9

Например, для частного случая краевых условий

$$z(a)=\alpha,\quad z(b)=\beta \tag{10.4.7}$$

задача называется краевой задачей первого рода или задачей с закрепленными концами (рис. 10.9). Пусть функция $q(x)$ удовлетворяет неравенству

$$q(x)\leqslant 0,\quad a\leqslant x\leqslant b. \tag{10.4.8}$$

Условие неположительности $q(x)$ обеспечивает существование и единственность решения некоторых краевых задач, в частности задачи с закрепленными концами. Справедливо следующее утверждение.

Теорема 10.5. *Пусть функции $q(x)$, $r(x)$ дважды непрерывно дифференцируемы на $[a, b]$; пусть также $q(x)\leqslant 0$. Тогда краевая задача* (10.4.5), (10.4.7) *имеет единственное решение $z(x)$ и $z(x)\in C^4[a, b]$.*

Перейдем от непрерывной задачи к дискретной, заменив производную в уравнении и производные в краевых условиях по формулам численного дифференцирования на сетке:

$$x_i=a+ih,\quad 0\leqslant i\leqslant m,\quad h=\frac{b-a}{m}.$$

Для внутренних узлов x_i, $1\leqslant i\leqslant m-1$, имеем

$$\frac{z(x_{i+1})-2z(x_i)+z(x_{i-1})}{h^2}+O(h^2)+q(x_i)z(x_i)=r(x_i),$$

для краевых узлов аппроксимируем первые производные с точностью до $O(h^2)$ (см. (10.1.5)), из (10.4.6) получаем

$$\alpha_0 z(x_0)+\alpha_1\frac{-z(x_2)+4z(x_1)-3z(x_0)}{2h}+O(h^2)=\alpha_2,$$

$$\beta_0 z(x_m)+\beta_1\frac{3z(x_m)+4z(x_{m-1})+z(x_{m-2})}{2h}+O(h^2)=\beta_2.$$

Если отбросить слагаемые порядка $O(h^2)$, то получим разностные уравнения (разностную схему)

$$\frac{z_{i+1}-2z_i+z_{i-1}}{h^2}+q_i z_i=r_i,\quad 1\leqslant i\leqslant m-1, \tag{10.4.9}$$

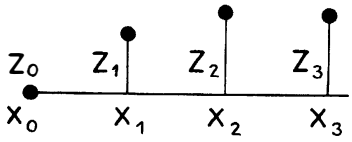

Рис. 10.10

$$\alpha_0 z_0+\frac{\alpha_1}{2h}(-z_2+4z_1-3z_0)=\alpha_2,$$
$$\beta_0 z_m+\frac{\beta_1}{2h}(3z_m-4z_{m-1}+z_{m-2}) \qquad (10.4.10)$$

— систему $(m+1)$-го уравнения с $m+1$ неизвестным z_0, z_1, ..., z_m. Здесь $q(x_i)=q_i$, $r(x_i)=r_i$.

Заметим, что на функциях $z(x)\in C^4[a, b]$ разностная схема (10.4.9), (10.4.10) аппроксимирует непрерывную краевую задачу с точностью $O(h^2)$ (отброшенные слагаемые).

Если решить разностные уравнения — определить z_i, $0\leqslant i\leqslant m$, то полученные значения z_i при выполнении условий устойчивости являются приближениями с точностью $O(h^2)$ к значениям $z(x_i)$ решения краевой задачи.

Например, пусть требуется решить краевую задачу

$$\frac{d^2z}{dx^2}-z=0,\quad z_0=0,\quad z(1)=1.$$

Выберем шаг сетки $h=1/3$ (рис. 10.10). Разностная схема запишется следующим образом:

$$z_0=0,$$
$$\frac{z_2-2z_1+z_0}{(1/3)^2}-z_1=0;$$
$$\frac{z_3-2z_2+z_1}{(1/3)^2}-z_2=0,$$
$$z_3=1.$$

Эта система уравнений легко решается. Находим: $z_0=0$, $z_1=0{,}2893$, $z_2=0{,}6107$, $z_3=1$. Сравнение с точным решением в узлах сетки $z(x_0)=0$, $z(x_1)=0{,}2889$, $z(x_2)=0{,}6102$, $z(x_3)=1$ указывает на хорошее приближение решения разностной схемы.

Преобразуем уравнения (10.4.10) так, чтобы исключить z_2 из первого уравнения и z_{m-2} — из второго (при этом естественно считать, что $\alpha_1\neq 0$, $\beta_1\neq 0$). Для исключения z_2 выразим z_2 через z_1, z_0 из (10.4.9). Имеем

$$z_2=r_1h^2-q_1h^2z_1+2z_1-z_0.$$

Подставляя правую часть равенства в (10.4.10), найдем

$$\alpha_0 z_0+\frac{\alpha_1}{2h}(-r_1h^2+q_1h^2z_1-2z_1+z_0+4z_1-3z_0)=\alpha_2,$$

откуда получаем

$$z_0=E_1z_1+D_1, \qquad (10.4.11)$$

где константы E_1, D_1 соответственно равны

$$E_1=\frac{2\alpha_1+\alpha_1q_1h^2}{2\alpha_1-2\alpha_0h},\quad D_1=-\frac{2\alpha_2h+\alpha_1r_1h^2}{2\alpha_1-2\alpha_0h}.$$

Для малых значений h знаменатель не равен нулю, так как $\alpha_1 \neq 0$.

Аналогично из (10.4.9) находим

$$z_{m-2} = r_{m-1}h^2 - q_{m-1}h^2 z_{m-1} + 2z_{m-1} - z_m.$$

Подставляя правую часть этого неравенства в (10.4.10), получаем

$$\beta_0 z_m + \frac{\beta_1}{2h}(3z_m - 4z_{m-1} + r_{m-1}h^2 - q_{m-1}h^2 z_{m-1} + 2z_{m-1} - z_m) = \beta_2.$$

Отсюда

$$z_m = Qz_{m-1} + S, \tag{10.4.12}$$

где константы Q, S соответственно равны

$$Q = \frac{2\beta_1 + \beta_1 q_{m-1}h^2}{2\beta_1 + 2\beta_0 h}; \quad S = \frac{2\beta_2 - \beta_1 r_{m-1}h^2}{2\beta_1 + 2\beta_0 h}.$$

Для малых значений h знаменатель не равен нулю, так как $\beta_1 \neq 0$.

Окончательно разностная схема краевой задачи представляет собой систему линейных алгебраических уравнений (10.4.11), (10.4.9), (10.4.12) относительно $z = (z_0, z_1, \ldots, z_m)$. Запишем ее в развернутой форме:

$$\begin{pmatrix} 1 & -E_1 & & & & \\ \frac{1}{h^2} & \left(-\frac{2}{h^2} + q_1\right) & \frac{1}{h^2} & & 0 & \\ & \ddots & \ddots & \ddots & & \\ & & \frac{1}{h^2} & \left(-\frac{2}{h^2} + q_i\right) & \frac{1}{h^2} & \\ & 0 & & \ddots & \ddots & \ddots \\ & & & \frac{1}{h^2} & \left(-\frac{2}{h^2} + q_{m-1}\right) & \frac{1}{h^2} \\ & & & 1 & -Q & \end{pmatrix} \times$$

$$\times \begin{pmatrix} z_0 \\ z_1 \\ \vdots \\ z_i \\ \vdots \\ z_{m-1} \\ z_m \end{pmatrix} = \begin{pmatrix} D_1 \\ r_1 \\ \ldots \\ r_i \\ \vdots \\ r_{m-1} \\ S \end{pmatrix}, \tag{10.4.13}$$

в матричной форме:

$$Az = b,$$

где A — матрица системы (10.4.13), b — вектор правых частей $b=(D_1, r_1, r_2, \ldots, r_{m-1}, S)$. Матрица A имеет ненулевые элементы, расположенные на трех диагоналях, т. е. это трехдиагональная матрица.

Рассмотренные методы решения линейных систем в гл. 9, ориентированные на матрицы общего вида, оказываются крайне неэффективными, если использовать их для решения систем с трехдиагональными матрицами. Например, метод исключения Гаусса, требующий для своей реализации $O(m^3)$, $m\to\infty$, арифметических операций, фактически будет обрабатывать основное время нулевые элементы, так как ненулевых элементов в матрице A всего $O(m)$.

Для решения систем линейных уравнений с трехдиагональными матрицами используется вариант исключения Гаусса, в котором принимают участие только ненулевые элементы A, так называемый *метод прогонки*. Число арифметических операций, необходимое для его реализации, оказывается равным $O(m)$. Следует отметить, что метод прогонки в системах линейных алгебраических уравнений — это дискретный вариант непрерывного метода прогонки, изложенного в п. 5.3.18.

10.4.2. Метод прогонки. Рассмотрим трехдиагональную систему уравнений более общего, чем (10.4.13), вида, а именно состоящую из (10.4.11), (10.4.12) и следующих уравнений:

$$A_i z_{i-1} - C_i z_i + B_i z_{i+1} = r_i, \quad 1 \leqslant i \leqslant m-1. \tag{10.4.14}$$

В (10.4.13) побочные диагонали являются константами, а здесь A_i, B_i могут зависеть от i.

Для вывода формул прогонки представим зависимость z_i от z_{i+1} по аналогии с (10.4.11) в виде

$$z_i = E_{i+1} z_{i+1} + D_{i+1}, \quad 0 \leqslant i \leqslant m-1, \tag{10.4.15}$$

где E_1, D_1 — известные константы, а $(E_2, \ldots, E_{m-1})$, $(D_2, \ldots, D_{m-1})$ — пока неизвестные константы. Из (10.4.15) получаем соотношения

$$z_{i-1} = E_i z_i + D_i = E_i(E_{i+1} z_{i+1} + D_{i+1}) + D_i =$$
$$= E_i E_{i+1} z_{i+1} + E_i D_{i+1} + D_i, \quad 1 \leqslant i \leqslant m-1. \tag{10.4.16}$$

Подставляя (10.4.15), (10.4.16) в (10.4.14), имеем

$$A_i E_i E_{i+1} z_{i+1} + A_i E_i D_{i+1} + A_i D_i - C_i E_{i+1} z_{i+1} -$$
$$- C_i D_{i+1} + B_i z_{i+1} - r_i = 0, \quad 1 \leqslant i \leqslant m-1.$$

Приравнивая в этом соотношении отдельно коэффициенты при z_{i+1} и свободные члены нулю, получаем

$$A_i E_i E_{i+1} - C_i E_{i+1} + B_i = 0,$$
$$A_i E_i D_{i+1} + A_i D_i - C_i D_{i+1} - r_i = 0,$$

откуда выразим E_{i+1}, D_{i+1} через E_i, D_i, а именно:

$$\begin{cases} E_{i+1}=\dfrac{B_i}{C_i-A_iE_i}, & 1\leqslant i\leqslant m-1, \\ D_{i+1}=\dfrac{A_iD_i-r_i}{C_i-A_iE_i}, & 1\leqslant i\leqslant m-1. \end{cases} \tag{10.4.17}$$

Формулы (10.4.17) — это формулы прямой прогонки. Они позволяют по заданным значениям E_1, D_1 последовательно определить (если знаменатели не обращаются в нуль) (E_2, D_2), (E_3, D_3), ..., (E_m, D_m).

Определив E_m, D_m, подставляем эти значения в (10.4.12), в результате получаем

$$z_m=Q(E_mz_m+D_m)+S.$$

Предполагая, что

$$(1-QE_m)\neq 0,$$

находим

$$z_m=\frac{QD_m+S}{1-QE_m}. \tag{10.4.18}$$

Обратная прогонка — это вычисление неизвестных z_i, $i=m-1$, $m-2$, ..., 0, по формуле (10.4.15). Вычисления можно осуществить, как только из (10.4.18) получено значение z_m.

Прогонка устойчива, если

$$|E_{i+1}|<1,\ \ 0\leqslant i\leqslant m-1, \tag{10.4.19}$$

в этом случае ошибки, возникающие в ходе вычислений, не накапливаются, т. е. не зависят от m. Этот факт следует из (10.4.15).

Теорема 10.6. *Прогонка осуществима и устойчива, если выполнены неравенства*

$$\begin{cases} A_i>0,\ B_i>0,\ C_i\geqslant A_i+B_i,\ \ 1\leqslant i\leqslant m-1, \\ 0\leqslant E_1<1,\ 0\leqslant Q<1. \end{cases} \tag{10.4.20}$$

Доказательство. Для E_1 неравенство (10.4.19) следует из условия теоремы. Для $i=1, 2, ..., m-1$ неравенство нулю знаменателя в формулах прогонки и (10.4.19) вытекает из следующей цепочки соотношений:

$$E_{i+1}=\frac{B_i}{C_i-A_iE_i}=\frac{B_i}{(C_i-A_i-B_i)+(A_i+B_i)-A_iE_i}=$$

$$=\frac{B_i}{(C_i-A_i-B_i)+B_i+(1-E_i)A_i}.$$

Действительно, поскольку числитель и все слагаемые знаменателя положительны, а знаменатель больше числителя для $i=1$,

утверждение теоремы верно для E_2. Аналогично можно показать, что оно верно для $i=3, \ldots, m-1$, а условие $0 \leqslant Q<1$ обеспечивает неравенство нулю знаменателя в (10.4.18), что и требовалось доказать.

10.4.3. Метод прогонки в разностной схеме краевой задачи. Разностная схема рассматриваемой краевой задачи (10.4.13) имеет следующие значения коэффициентов в формулах прогонки:

$$A_i=1/h^2, \quad B_i=1/h^2, \quad C_i=2/h^2-q_i.$$

Если $\alpha_1=0$, $\beta_1=0$ (что соответствует задаче с закрепленными концами), то

$$E_1=0, \quad D_1=\alpha_2/\alpha_0, \quad Q=0, \quad S=\beta_2/\beta_0.$$

В силу неравенства (10.4.8) прогонка для разностной схемы устойчива, поскольку неравенства (10.4.20) все выполнены.

Если рассматривается краевая задача общего вида и $\alpha_1 \neq 0$, $\beta_1 \neq 0$, то для достаточно малых h из формул для E_1 и Q находим

$$E_1=1+\frac{\alpha_0}{\alpha_1}h+O(h^2),$$

$$Q=1-\frac{\beta_0}{\beta_1}h+O(h^2),$$

откуда следует, что прогонка устойчива, если

$$\alpha_0/\alpha_1<0, \quad \beta_0/\beta_1>0. \tag{10.4.21}$$

Выше уже упоминалось, что разностная схема аппроксимирует краевую задачу с точностью $O(h^2)$. Но аппроксимация и устойчивость дают сходимость со скоростью, определяемой порядком аппроксимации (см. п. 10.1.6) решения разностного уравнения к точному решению краевой задачи. Теперь можно сформулировать следующее утверждение.

Теорема 10.7. *Пусть* $q(x)$, $r(x) \in C^2[a, b]$, $q(x) \leqslant 0$ *на интервале* $[a, b]$, *пусть выполнено условие* (10.4.21). *Тогда решение* z_i *разностной схемы* (10.4.13), *полученное методом прогонки, сходится к точному решению краевой задачи* (10.4.5), (10.4.6) *и имеет место оценка*

$$\max_{0 \leqslant i \leqslant m} |z_i-z(x_i)| \leqslant O(h^2), \quad h \to 0.$$

10.4.4. Краевая задача для системы линейных дифференциальных уравнений. Система линейных дифференциальных уравнений, как правило, возникает после этапа линеаризации исходной нелинейной системы в окрестности некоторого решения. Поэтому, решая линейные уравнения, следует всегда помнить о их происхождении. Например, норма решения системы линейных уравнений во многих случаях должна быть малой, иначе рассматриваемая линейная модель окажется неверной (большая погрешность математической модели).

Обширный класс технических задач, приводящий к краевым задачам для систем линейных уравнений, представляют собой

проблемы автоматического управления. Многие из этих задач можно записать в виде

$$\ddot{q}_i = F_i(\dot{q}_i, q_i, u_r(t)), \quad 1 \leqslant i \leqslant n, \quad 1 \leqslant r \leqslant m.$$

Здесь q_i — обобщенные координаты управляемой системы, t — время $(0 \leqslant t \leqslant T)$, $\dot{q}$ — производная по времени, $u_r(t)$ — управляющие воздействия на систему. Простейшей задачей управления является следующая: при заданных управляющих воздействиях $u_r(t)$ на систему найти траекторию $q_i(t)$, переводящую систему из заданного начального положения

$$q_i(0) = q_i^{(0)}, \quad 1 \leqslant i \leqslant n,$$

в заданное конечное

$$q_i(T) = q_i^{(1)}, \quad 1 \leqslant i \leqslant n.$$

Например, задача определения траектории управляемого снаряда принадлежит этому классу. Действительно, в плоской модели полета обобщенные координаты $q_1 = x$, $q_2 = y$, уравнения движения в линеаризованном виде записываются в форме

$$\begin{cases} \ddot{x} = u_1(t)\dot{x} + u_2(t)y + u_3(t), \\ \ddot{y} = u_1(t)\dot{y} + u_2(t)y - g, \end{cases}$$

где g — ускорение свободного падения, $u_1(t)$, $u_2(t)$, $u_3(t)$ — управляющие воздействия, определяемые тягой двигателя и сопротивлением воздуха. Начальное положение — это координата старта, конечное — координата цели (рис. 10.11). Таким образом, краевые условия

$$x(0) = x^{(0)}, \quad x(T) = x^{(1)};$$
$$y(0) = y^{(0)}, \quad y(T) = y^{(1)}.$$

Если условие в начальный момент $t = 0$ дополнить еще двумя, определяющими угол θ запуска снаряда $(\theta = \operatorname{arctg}(v_1/v_0))$,

$$\dot{x}(0) = v_0, \quad \dot{y}(0) = v_1,$$

то получим на интервале $0 \leqslant t \leqslant T$ задачу Коши, которая имеет единственное решение, но, вообще говоря, траектория $(x(t), y(t))$ не проходит через цель при $t = T$ (рис. 10.11).

Если теперь будем задавать различные значения (v_0, v_1), т. е. изменять угол стрельбы θ, то будем получать различные решения задачи Коши, и, возможно, среди значений (v_0, v_1) найдем такой угол θ, при котором траектория с заданной точностью проходит через цель.

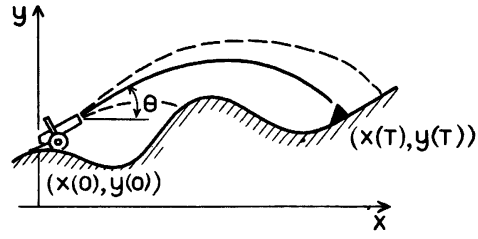

Рис. 10.11

Метод решения краевых задач путем сведения их к задаче Коши с последующим поиском начальных значений

таких, чтобы выполнялись краевые условия, называется методом стрельбы.

Рассмотрим систему линейных дифференциальных уравнений

$$\frac{dy_i}{dx}=\sum_{j=1}^{n} a_{i,j}(x)\,y_j+f_i(x),\quad 1\leqslant i\leqslant n,$$

с линейными краевыми условиями

$$\sum_{j=1}^{n} G_{i,j}\,y_j(a)+\sum_{j=1}^{n} D_{i,j}\,y_j(b)=c_i,\quad 1\leqslant i\leqslant n,$$

где $a_{i,j}(x)$, $f_i(x)$ — элементы переменных матрицы $A(x)$, вектора $f(x)$; $G_{i,j}$, $D_{i,j}$, c_i — элементы постоянных матриц G, D, вектора c. В матричной записи краевая задача имеет вид

$$\frac{dy}{dx}=A(x)\,y+f, \tag{10.4.22}$$

$$Gy(a)+Dy(b)=c. \tag{10.4.23}$$

Для примера запишем матрицы G и D вектор c в следующих условиях:

$$y_1(a)=0;\quad y_3(a)=1;\quad y_2(b)=2;\quad y_4(b)=3.$$

Имеем

$$G=\begin{pmatrix}1&0&0&0\\0&0&0&0\\0&0&1&0\\0&0&0&0\end{pmatrix};\quad D=\begin{pmatrix}0&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&0&1\end{pmatrix};\quad c=\begin{pmatrix}0\\2\\1\\3\end{pmatrix}.$$

Для условий

$$y_1(a)=0,\quad y_3(a)=1,\quad y_1(b)=2,\quad y_3(b)=3$$

имеем

$$G=\begin{pmatrix}1&0&0&0\\0&0&0&0\\0&0&1&0\\0&0&0&0\end{pmatrix};\quad D=\begin{pmatrix}0&0&0&0\\1&0&0&0\\0&0&0&0\\0&0&1&0\end{pmatrix};\quad c=\begin{pmatrix}0\\2\\1\\3\end{pmatrix}.$$

Метод стрельбы в приложении к краевой задаче (10.4.22) состоит из следующих этапов:

1. Задают начальное приближение для вектора

$$y(a)=y^{(0)}(a). \tag{10.4.24}$$

2. Решают каким-либо численным методом задачу Коши (10.4.22), (10.4.24) на интервале $a\leqslant x\leqslant b$, в результате получают вектор $y^{(0)}(b)$.

3. Вычисляют

$$r^{(0)}=Gy^{(0)}(a)+Dy^{(0)}(b)-c.$$

Вектор $r=Gy(a)+y(b)-c$ называется *вектором невязки краевых условий*, этот вектор зависит от $y^{(0)}(a)$. Вычислим норму вектора невязки — число S:

$$S(y^{(0)}(a))=\|Gy^{(0)}(a)+Dy^{(0)}(b)-c\|.$$

4. Если $S(y^{(0)}(a))=0$, то краевая задача решена, если нет, то изменяют начальное приближение (10.4.24) так, чтобы целевая функция $S(y^{(0)}(a))$ (см. гл. 9) уменьшилась.

Процедуру 1—4 продолжают до тех пор, пока S не станет меньше заданной точности.

Вопросы существования точного решения краевой задачи устойчивости метода стрельбы выходят за рамки этой книги. Практически алгоритм метода стрельбы следует применять только для матриц $A(x)$, собственные значения которых на нормированном интервале $0\leqslant x\leqslant 1$ имеют небольшие модули, т. е. $|\lambda_i|\simeq 1$.

Метод стрельбы является при определенной дискретизации краевой задачи фактически одним из способов решения полученной системы линейных алгебраических уравнений. В общем случае способ решения полученной системы линейных уравнений определяет характер поведения собственных значений матрицы A (система уравнений, например, может быть жесткой).

Пусть интервал $[a, b]$ разбит постоянным шагом h узлами $x_j=a+jh$, $0\leqslant j\leqslant m$, $h=(b-a)/m$. Точное решение $y(x)$ в узлах представляет собой матрицу неизвестных

$$y_i(x_j),\ 1\leqslant i\leqslant n,\ 0\leqslant j\leqslant m,$$

т. е. всего $(m+1)n$ неизвестных. Заменим краевую задачу (10.4.22), (10.4.23) разностной схемой с аппроксимацией второго порядка. Для внутренних узлов

$$\frac{y_{j-1}-y_{j-1}}{2h}=A(x_j)y_j+f(x_j),\ 1\leqslant j\leqslant m-1,$$

краевое условие дает

$$Gy_0+Dy_m=c.$$

Добавим для замыкания разностной схемы уравнение

$$\sigma\left(\frac{-y_2+4y_1-3y_0}{2h}-A(x_0)y_0+f(x_0)\right)+$$
$$+(1-\sigma)\left(\frac{3y_m-4y_{m-1}+y_{m-2}}{2h}-A(x_m)y_m+f(x_m)\right)=0,$$

где $0\leqslant\sigma\leqslant 1$. Полученная система разностных уравнений является системой линейных алгебраических уравнений $(m+1)n$-го порядка относительно неизвестных $y_{i,j}$, $1\leqslant i\leqslant n$, $0\leqslant j\leqslant m$. Решив эту систему, получим приближенные значения $y_{i,j}$ к точному решению в узлах $y_i(x_j)$.

Имеются разнообразные методы решения таких разностных схем, с которыми более подробно можно познакомиться в [25].

10.4.5. Применение программы B7A0. Для демонстрации применения программы B7A0 решения краевой задачи рассмотрим систему уравнений

$$\frac{dy_1}{dx} = xy_1 + y_2 - \sin x,$$
$$\frac{dy_2}{dx} = y_1 - x^2 y_2 + y_3,$$
$$\frac{dy_3}{dx} = y_2 + (\cos x) y_3 + 1$$

на интервале $0 \leqslant x \leqslant 1$ с условиями

$$y_1(0) + y_2(0) = 0,$$
$$y_1(0) - y_3(1) = 1,$$
$$y_2(1) + y_3(1) = 2.$$

Пусть требуется точность вычислений $\varepsilon = 10^{-2}$. Программа может иметь следующий вид:

```
      REAL A,B,E,G(3,3),D(3,3),C(3),X(32),Y(3,32),W(1610)
      INTEGER N,M,N1,L,J(227),K,I
      EXTERNAL Q,F
      DATA A,B /0.,1./,E/1.E-2/
      DATA G/1.,1.,0.,1.,0.,0.,0.,0.,0./
      DATA D/0.,0.,0.,0.,0.,1.,0.,—1.,1./
      DATA C/0.,1.,2./
      DATA X/0.,0.2,0.4,0.6,0.8,1.0/
      DATA N,M,N1,L,K,I/3,32,6,1610,227,110/
C     ОБРАЩЕНИЕ К ПРОГРАММЕ B7A0
      CALL B7A0(A,B,N,E,Q,F,G,D,C,M,X,Y,N1,W,L,J,K,I)
C     ВЫВОД НА ТЕРМИНАЛ УЗЛОВ X И ВЕКТОРА Y
      WRITE (5,1) (X(I)(Y(J,I)J=1,3)I=1,N1)
1     FORMAT (2X,'X=',F6.3,3E13.6)
      END
C     ВНЕШНЯЯ ПОДПРОГРАММА, ВЫЧИСЛЯЮЩАЯ
C     МАТРИЦУ СИСТЕМЫ
      SUBROUTINE Q(X,Z)
      REAL X,Z(3,3)
      Z(1,1)=X
      Z(1,2)=1.
      Z(1.3)=0.
      Z(2,1)=1.
      Z(2,2)=-X*X
      Z(2,3)=0.
      Z(3,1)=0.
      Z(3,2)=1.
      Z(3,3)=COS(X)
      RETURN
      END
C     ВНЕШНЯЯ ПОДПРОГРАММА, ВЫЧИСЛЯЮЩАЯ F(X)
```

```
      SUBROUTINE F(X,R)
      REAL X,R(3)
      R(1)=-SIN(X)
      R(2)=0.
      R(3)=1.
      RETURN
      END
```

10.4.6. Краевая задача для систем нелинейных дифференциальных уравнений. Рассмотрим систему нелинейных дифференциальных уравнений

$$\frac{dy_i}{dx}=f_i(x,\ y_1,\ \ldots,\ y_n),\quad 1\leqslant i\leqslant n, \tag{10.4.25}$$

на интервале $a\leqslant x\leqslant b$ с краевыми условиями

$$\begin{aligned} y_i(a)&=y^{(0)},\quad 1\leqslant i\leqslant k,\ k<n,\\ y_i(b)&=y^{(1)},\quad k+1\leqslant i\leqslant n. \end{aligned} \tag{10.4.26}$$

Подход к численному решению задачи (10.4.25), (10.4.26) аналогичен изложенному выше для систем линейных уравнений.

Метод стрельбы состоит в том, что вектор $y(a)$ полностью определяется в точке $x=a$, т. е. дополнительно задаются

$$y_i(a)=y_i^{(0)},\quad k+1\leqslant i\leqslant n,$$

нулевое приближение для недостающих до полного вектора координат. Затем каким-либо численным методом решается задача Коши для системы дифференциальных уравнений (10.4.25) и определяется вектор $\tilde{y}(b)$. Следующим этапом является минимизация целевой функции, например

$$S(y_{k+1}^{(0)},\ \ldots,\ y_n^{(0)})=\sqrt{\sum_{i=k+1}^{n}(y_i^{(1)}-\tilde{y}_i(b))^2},$$

каким-либо методом нелинейной минимизации по переменным $y_{k+1}^{(0)},\ \ldots,\ y_n^{(0)}$.

Метод стрельбы обычно устойчив, если на точном решении якобиан

$$\frac{\partial f_i}{\partial y_j}(x,\ y_1(x),\ \ldots,\ y_n(x))$$

имеет собственные значения $\lambda_i(x)$ на нормированном интервале $0\leqslant x\leqslant 1$ такие, что

$$|\lambda_i(x)|\simeq 1.$$

Другой подход состоит в переходе к разностной схеме и решению полученной нелинейной системы уравнений одним из способов, изложенным в гл. 9.

На сетке $x_i=a+ih$, $0\leqslant i\leqslant m$, $m=(b-a)h$ заменим дифференциальное уравнение конечно-разностным выражением. В векторной записи имеем во внутренних узлах

$$\frac{y_{j+1}-y_{j-1}}{2h}=f(x_j,\ y_j),\quad 1\leqslant j\leqslant m-1,$$

в краевых узлах

$$\frac{-y_2+4y_1-3y_0}{2h}=f(x_0,\ y_0),$$

$$\frac{3y_m-4y_{m-1}+y_{m-2}}{2h}=f(x_m,\ y_m).$$

Полученная система нелинейных уравнений совместно с условиями (10.4.26), если она разрешима, дает приближения $y_{i,j}$, $1\leqslant i\leqslant n$, $0\leqslant j\leqslant m$, к точному решению исходной краевой задачи в узлах $y_i(x_j)$.

10.4.7. Применение программы B7A1. Рассмотрим краевую задачу для системы уравнений

$$\frac{dy_1}{dx}=xy_1+\sin y_2-\sin x,$$

$$\frac{dy_2}{dx}=y_1-x^2y_2+\cos y_3,$$

$$\frac{dy_3}{dx}=\sin y_2+(\cos x)y_3+1$$

на интервале $0\leqslant x\leqslant 1$ с условиями

$$y_1(0)=y_2(0)=0,\ \ y_3(1)=1.$$

Пусть требуемая точность вычислений $\varepsilon=10^{-2}$. Программа может иметь следующий вид:

```
      REAL U(3,2),V(3,2),A,B,E,X(32),Y(3,32),W(1712)
      INTEGER N,M,N1,L,J(247),K,I
      EXTERNAL F
      DATA U(1,1),U(2,1),U(3,1)/0.,0.,0./
      DATA U(1,2),U(2,2),U(3,2)/0.,0.,1./
      DATA V(1,1),V(2,1),V(3,1)/0.,0.,1./
      DATA V(1,2),V(2,2),V(3,2)/1.,1.,0./
      DATA X/0.,0.2,0.4,0.6,0.8,1.0/,A,B,E/0.,1.,1.E—2/
      DATA N,M,N1,L,K,I/3,32,6,1712,247,110/
C     ОБРАЩЕНИЕ К ПРОГРАММЕ B7A1
      CALL B7A1(U,V,N,A,B,E,F,M,X,Y,N1,W,L,J,K,I)
C     ВЫВОД НА ТЕРМИНАЛ УЗЛОВ X И ВЕКТОРА Y
      WRITE (5,1) (X(I)(Y(J,I)J=1,3)I=1,N1)
1     FORMAT (2X,'X=',F6.3,3E13.6)
      END
C     ВНЕШНЯЯ ПОДПРОГРАММА, ВЫЧИСЛЯЮЩАЯ
C     F(X,Y)
      SUBROUTINE F(X,Y,R)
      REAL X,Y(3),R(3)
      R(1)=X*Y(1)+SIN(Y(2))-SIN(X)
      R(2)=Y(1)-X*X*Y(2)+COS(Y(3))
      R(3)=SIN(Y(2))+(COS(X))*Y(3)+1.
      RETURN
      END
```

Глава 11

УРАВНЕНИЯ С ЧАСТНЫМИ ПРОИЗВОДНЫМИ

● 11.1. Введение

11.1.1. Ограниченность применения аналитических методов. В гл. 5 рассматривались аналитические методы решения уравнений с частными производными. В основе успеха применения этих методов лежала возможность приведения уравнения с частными производными к системе обыкновенных дифференциальных уравнений. Обыкновенные дифференциальные уравнения также должны быть достаточно простыми, чтобы их решения можно было записать в известных функциях, т. е. таких, для которых имеются таблицы значений или программы вычисления значений.

Как правило, аналитические методы в линейных уравнениях связаны с разделением переменных (методом Фурье). Использование этого метода встречает большие трудности, если область независимых переменных, где ищется решение, отличается от простейших (прямоугольник, круг и т. п.). Вторым препятствием для применения метода Фурье является зависимость коэффициентов линейного уравнения от времени и пространственных переменных. Например, зависимость коэффициента $a=a(t, x)$ в волновом уравнении с функцией $a(t, x)$ достаточно общего вида

$$\frac{\partial^2 u}{\partial t^2}=a^2(t, x)\frac{\partial^2 u}{\partial x^2}$$

уже не позволит разделить переменные.

Аналитические методы решения нелинейных уравнений используют глубокие внутренние свойства дифференциальных уравнений, их применение требует высокой математической квалификации, а виды уравнений, доступных решению, весьма ограниченны. Поэтому без большого преувеличения можно сказать, что нелинейность в дифференциальном уравнении не позволит инженеру использовать аналитические методы в практической работе.

Итак, имеется три основных «источника неприятностей» для аналитических методов в уравнениях с частными производными:

1) сложная область в линейных уравнениях с постоянными коэффициентами;

2) переменные коэффициенты в линейных уравнениях;

3) нелинейность.

Примером трудностей первого типа является уравнение Лапласа

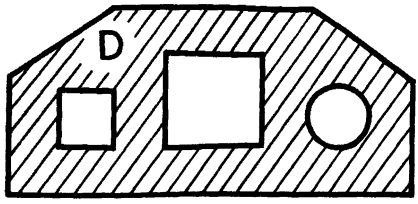

Рис. 11.1

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=0$$

в области D (рис. 11.1) с каким-либо условием на границе D.

Пример второго типа — уравнение поперечных колебаний мембраны

$$\frac{\partial^2 u}{\partial t^2}=\frac{T}{\rho(x, y)}\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right),$$

где T — коэффициент натяжения, $\rho(x, y)$ — плотность мембраны. Неоднородность мембраны $\rho(x, y)\neq\text{const}$ и есть источник трудностей в аналитических методах.

Примером трудностей третьего типа служит уравнение теплопроводности в случае, когда коэффициент теплопроводности зависит от температуры:

$$\frac{\partial u}{\partial t}=a(u)\frac{\partial^2 u}{\partial x^2}.$$

Отмеченные ограничения применения аналитических методов привели, особенно с развитием вычислительной техники, к широкому распространению численных методов решения уравнений с частными производными. Опыт решения многих сложных задач науки и техники численными методами подтверждает их эффективность.

11.1.2. Дискретизация. Первым этапом в численном решении дифференциальных уравнений, как обыкновенных (см. гл. 10), так и с частными производными, является переход от непрерывной задачи к дискретной.

Способ дискретизации задачи — это «визитная карточка» численного метода. Получаемые после дискретизации конечномерные задачи могут быть похожи для разных методов и решаться практически одинаково. Рассмотрим три метода дискретизации, которые связаны с наиболее популярными методами решения уравнений с частными производными, а именно: *вариационные методы*; *методы прямых*; *конечно-разностные методы.*

11.1.3. Вариационные методы. В основе вариационных методов лежит переход от задачи для уравнения с частными производными к задаче минимизации функционала. Этот прием обобщает идею перехода от решения уравнений к задаче минимизации целевой функции, построенной по исходным уравнениям (гл. 9).

Функционалом называется оператор, который ставит число в соответствие функции.

Примером функционала является оператор

$$I(u)=\iint\limits_D\left(1+\left(\frac{\partial u}{\partial x}\right)^2+\left(\frac{\partial u}{\partial y}\right)^2\right)^{1/2}dx\,dy,$$

который любой функции $u(x, y)$, непрерывно дифференцируемой в области D, т. е. $u(x, y) \in C^1[D]$, ставит в соответствие ее площадь поверхности.

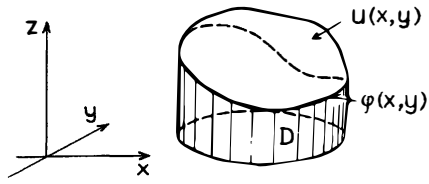

Рис. 11.2

Далее будем рассматривать функционал вида

$$I(u)=\iint\limits_D F\left(x, y, u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}\right) dx\,dy \qquad (11.1.1)$$

на функциях $u(x, y) \in C^1[D]$ таких, что на границе Γ области D функция u принимает заданные значения (рис. 11.2)

$$u(x, y)=\varphi(x, y), \quad (x, y) \in \Gamma. \qquad (11.1.2)$$

Пусть функционал $I(u)$ достигает на функции $u(x, y)$ своего минимального $\bar{I}$ значения $I(\bar{u})$. Это означает, что на «близких» функциях к $\bar{u}(x, y)$ функционал имеет значения, не меньшие $\bar{I}$. Близкие функции к $\bar{u}(x, y)$ определяются следующим образом:

$$u(x, y)=\bar{u}(x, y)+\alpha\eta(x, y),$$

где α — малые числа, $\eta(x, y) \in C^1[D]$,

$$\eta(x, y)=0, \quad (x, y) \in \Gamma.$$

На близких функциях можно определить функционал (11.1.1):

$$I(\alpha)=\iint\limits_D F\left(x, y, \bar{x}+\alpha\eta, \frac{\partial \bar{u}}{\partial x}+\alpha\frac{\partial \eta}{\partial x}, \frac{\partial \bar{u}}{\partial y}+\alpha\frac{\partial \eta}{\partial y}\right) dx\,dy.$$

Разложим $I(\alpha)$ в ряд Тейлора по α с центром в нуле. Получим

$$I(\alpha)=\bar{I}+\frac{dI}{d\alpha}(0)\alpha+\frac{1}{2}\frac{d^2I}{d\alpha^2}(0)\alpha^2+\ldots$$

Первой вариацией функционала $I(u)$ называется величина

$$\delta I=\frac{dI}{d\alpha}(0)$$

второй вариацией — величина

$$\delta^2 I=\frac{d^2I}{d\alpha^2}(0).$$

Теорема 11.1. *Если функционал $I(u)$ достигает на функции $\bar{u}(x, y)$ минимального значения, то*

$$\delta I=0, \quad \delta^2 I \geqslant 0.$$

Доказательство этой теоремы аналогично соответствующему доказательству для минимума функции одной переменной $I(u)$ (рис. 11.3). Запишем разложение $I(\alpha)$ для функционала (11.1.1)

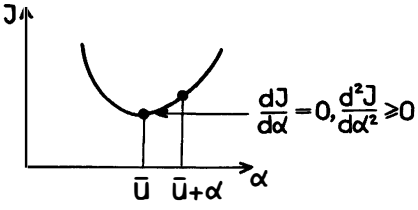

Рис. 11.3

в предположении, что функция F дважды непрерывно дифференцируема по аргументам u, $\frac{\partial u}{\partial x}$, $\frac{\partial u}{\partial y}$. Имеем

$$I(\alpha)=\int\limits_D\int F\left(x,\ y,\ \bar{u},\ \frac{\partial\bar{u}}{\partial y},\ \frac{\partial\bar{u}}{\partial y}\right)dxdy+$$

$$+\alpha\int\limits_D\int\left[\left(\frac{\partial\bar{F}}{\partial u}\right)\eta+\left(\frac{\partial\bar{F}}{\partial u'_x}\right)\left(\frac{\partial\eta}{\partial x}\right)+\left(\frac{\partial\bar{F}}{\partial u'_y}\right)\left(\frac{\partial\eta}{\partial y}\right)\right]dxdy+O(\alpha^2).$$

Отсюда определяется первая вариация функционала:

$$\delta I=\int\limits_D\int\left[\left(\frac{\partial\bar{F}}{\partial u}\right)\eta+\left(\frac{\partial\bar{F}}{\partial u'_x}\right)\left(\frac{\partial\eta}{\partial x}\right)+\left(\frac{\partial\bar{F}}{\partial u'_y}\right)\left(\frac{\partial\eta}{\partial y}\right)\right]dxdy.$$

Здесь черта сверху означает, что выражение берется на функции $\bar{u}(x, y)$. Преобразуем δI с учетом формулы интегрирования по частям в двумерной области (формула Грина):

$$\int\limits_D\int\frac{\partial F_1}{\partial x}F_2\,dxdy=-\int\limits_D\int F_1\frac{\partial F_2}{\partial x}dxdy+\int\limits_\Gamma F_1F_2\cos(n,\ x)\,d\Gamma,$$

где n — нормаль к границе. Аналогичная формула справедлива для $\frac{\partial}{\partial y}$. Получим

$$\delta I=\int\limits_D\int\left[\left(\frac{\partial\bar{F}}{\partial u}\right)-\frac{\partial}{\partial x}\left(\frac{\partial\bar{F}}{\partial u'_x}\right)-\frac{\partial}{\partial y}\left(\frac{\partial\bar{F}}{\partial u'_y}\right)\right]\eta(x,\ y)\,dxdy+$$

$$+\int\limits_\Gamma\left[\left(\frac{\partial\bar{F}}{\partial u'_x}\right)\cos(n,\ x)+\left(\frac{\partial\bar{F}}{\partial u'_y}\right)\cos(n,\ y)\right]\eta(x,\ y)\,d\Gamma.$$

Но последнее слагаемое в правой части равно нулю, поскольку $\eta(x, y)=0$ при $(x, y)\in\Gamma$. Окончательно первая вариация функционала принимает вид

$$\delta I=\int\limits_D\int\left[\left(\frac{\partial\bar{F}}{\partial u}-\frac{\partial}{\partial x}\left(\frac{\partial\bar{F}}{\partial u'_x}\right)-\frac{\partial}{\partial y}\left(\frac{\partial\bar{F}}{\partial u'_y}\right)\right]\eta(x,\ y)\,dxdy.$$

Необходимое условие минимума $\delta I=0$ означает, что интеграл в правой части должен обращаться в нуль для любой функции $\eta(x, y)$, обращающейся в нуль на границе Γ, но отсюда следует (это можно строго доказать), что

$$\frac{\partial\bar{F}}{\partial u}-\frac{\partial}{\partial x}\left(\frac{\partial\bar{F}}{\partial u'_x}\right)-\frac{\partial}{\partial y}\left(\frac{\partial\bar{F}}{\partial u'_y}\right)=0 \qquad (11.1.3)$$

на функции $\bar{u}(x, y)$.

Уравнение (11.1.3) — это уравнение Эйлера, оно является необходимым условием минимума функционала (11.1.1) с условием

(11.1.2). Заметим, что (11.1.3) — уравнение с частными производными, а условие (11.1.2) — условие Дирихле, которое служит для однозначного определения решения (11.1.3). В основе вариационного метода лежит следующее утверждение.

Теорема 11.2. *Пусть функция* $\bar{u}(x, y)$ *доставляет минимум функционалу* (11.1.1) *на функциях* $u(x, y) \in C^2[D]$, *удовлетворяющих на границе условию* $u(x, y) = \varphi(x, y)$. *Тогда* $\bar{u}(x, y)$ — *решение дифференциального уравнения* (11.1.3) *с условием* (11.1.2).

Справедливо и обратное утверждение (при условии, что $\delta^2 I \geqslant 0$): решение $\bar{u}(x, y)$ задачи (11.1.3), (11.1.2) минимизирует функционал I.

Вариационным методом решения краевой задачи (11.1.3), (11.1.2) называется метод сведения ее к поиску минимума соответствующего функционала, для которого данное уравнение является уравнением Эйлера.

Следует отметить, что собственно вариационный метод не является численным методом, он широко применяется и в аналитических построениях. Как видно из изложенного выше, мы еще не ввели дискретизацию задачи.

Приведем примеры уравнений (11.1.3) и соответствующих функционалов:

$$1)\ \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0;\quad I(u) = \iint\limits_D \left[\left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial u}{\partial y} \right)^2 \right] dxdy;$$

$$2)\ \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = u;\quad I(u) = \iint\limits_D \left[\left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial u}{\partial y} \right)^2 + u^2 \right] dxdy;$$

$$3)\ \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y);\quad I(u) = \iint\limits_D \left[\left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial u}{\partial y} \right)^2 - 2uf \right] dxdy.$$

Последний пример определяет функционал задачи Дирихле для уравнения Пуассона. Для приведенных функционалов можно показать, что $\delta^2 I \geqslant 0$ на любой функции $u(x, y) \in C^2[D]$.

Функционал может содержать производные от $u(x, y)$ более высокой степени, нежели первая. Например, нагруженная пластина с заделанным краем

$$u(x, y) = 0,\ \frac{\partial u}{\partial n} = 0,\ (x, y) \in \Gamma,$$

описывается уравнением

$$\frac{\partial^4 u}{\partial x^4} + 2\frac{\partial^4 u}{\partial x^2 dy^2} + \frac{\partial^4 u}{\partial y^4} = f(x, y),$$

где $f(x, y)$ — нагрузка на пластину. Соответствующий функционал

$$I(u) = \iint\limits_D \frac{1}{2} \left[\left(\frac{\partial^2 u}{\partial x^2} \right)^2 + 2\frac{\partial^2 u}{\partial x \partial y} + \left(\frac{\partial^2 u}{\partial y^2} \right)^2 - 2uf \right] dxdy.$$

Будем предполагать, что задано однородное краевое условие $u(x, y) = 0$, $(x, y) \in \Gamma$; если это не так, то замена переменных

$v=u-\varphi(x, y)$ приводит к однородному краевому условию. Будем предполагать, что функционал $I(u)$ имеет положительно определенную вторую вариацию, т. е. $\delta^2 I(u)>0$ для любых $u\neq 0$, на которых $I(u)$ определен.

Отличительной чертой вариационного метода является более слабые требования к гладкости решения. Функционал $I(u)$ в примерах определен на функциях, имеющих первые производные, квадратично интегрируемые в области D. Таким образом, принадлежность $u(x, y)\in C^2[D]$ не требуется. Это важно в тех задачах, где граница D имеет угловые точки (например, прямоугольник) и, как правило, вторые производные решения в углах разрывны.

11.1.4. Дискретизация Ритца. Перенесем дискретизацию непрерывной задачи для уравнения (11.1.3) с условием (11.1.2) на эквивалентную вариационную задачу

$$\min_{u\in H} I(u), \tag{11.1.4}$$

где H — пространство функций $u(x, y)$, заданных на области D, для которых определен функционал $I(u)$. Будем предполагать, что решение (11.1.4) $u(x, y)$ существует, $u(x, y)\in H$; если $u(x, y)\in C^2[D]$, то это решение дифференциального уравнения (11.1.3) с условием (11.1.2). В том случае, когда решение (11.1.4) $u(x, y)$ не принадлежит $C^2[D]$, его называют *слабым решением* исходной краевой задачи.

Пусть функции $e_i(x, y)$

$$\{e_1, e_2, \ldots, e_m, \ldots\}$$

образуют базис пространства H.

Метод дискретизации Ритца состоит в том, что вариационная задача (11.1.4) решается не на всем пространстве H, а только на m-мерном подпространстве, состоящем из всевозможных линейных комбинаций элементов $\{e_1, e_2, \ldots, e_m\}$, т. е. решение $u(x, y)$ задачи (11.1.4) ищется в виде

$$u_m=\sum_{i=1}^{m}\alpha_i e_i, \tag{11.1.5}$$

где α_i, $1\leqslant i\leqslant m$ — неизвестные коэффициенты. При этом задача (11.1.4) переходит в дискретную задачу

$$\min_{\alpha} I\left(\sum_{i=1}^{m}\alpha_i e_i\right)$$

минимизации функционала на множестве векторов $\alpha=(\alpha_1, \ldots, \alpha_m)$ размерности m. Необходимое условие минимума

$$\frac{\partial I}{\partial \alpha_i}=0, \quad 1\leqslant i\leqslant m, \tag{11.1.6}$$

дает m уравнений для определения α_m. Если система уравнений (11.1.6) разрешима, то полученное решение α определяет u_m по

формуле (11.1.5) — приближение к точному решению исходной задачи. Очевидно, что u_m дает оценку точного минимального значения функционала

$$\min_{u \in H} I(u) \leqslant I(u_m).$$

В полном пространстве H точное решение (11.1.4) получается предельным переходом

$$u = \lim_{m \to \infty} u_m.$$

Для примера рассмотрим нелинейную краевую задачу

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = (u-1)^3$$

в области $D = \{0 \leqslant x \leqslant 1,\ 0 \leqslant y \leqslant 1\}$ с условием $u(\Gamma) = 0$. Соответствующий функционал

$$I(u) = \int_0^1 \int_0^1 \left[\left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial u}{\partial y} \right)^2 + \frac{(u-1)^4}{4} \right] dxdy.$$

Дискретизацию по методу Ритца проведем с помощью тригонометрического базиса. Ищем решение в виде

$$u_m(x, y) = \sum_{p,q=1}^{m} \alpha_{p,q} \sin p\pi x \sin q\pi y. \tag{11.1.7}$$

Для иллюстрации ограничимся одним слагаемым

$$u_1(x, y) = \alpha_{1,1} \sin \pi x \sin \pi y.$$

Найдем коэффициент $\alpha_{1,1}$ из условия

$$\min_{\alpha_{1,1}} I(\alpha_{1,1}) = \int_0^1 \int_0^1 \left[\alpha_{1,1}^2 \pi^2 \cos^2 \pi x \sin^2 \pi y + \right.$$

$$\left. + \alpha_{1,1}^2 \pi^2 \sin^2 \pi x \cos^2 \pi y + \frac{1}{4} (\alpha_{1,1} \sin \pi x \sin \pi y - 1)^4 \right] dxdy =$$

$$= \frac{9}{256} \alpha_{1,1}^4 - \frac{16}{(3\pi)^2} \alpha_{1,1}^3 + \left(\frac{3}{2} + \frac{\pi^2}{2} \right) \alpha_{1,1}^2 - \frac{4}{\pi^2} \alpha_{11} + \frac{1}{4}.$$

Определим точку минимума, отбросив слагаемые выше 2-й степени $\alpha_{1,1}$. Получим

$$\alpha_{1,1} \cong \frac{4}{\pi^2 (3 + \pi^2)} \cong 3{,}15 \cdot 10^{-2}.$$

Погрешность аппроксимации дифференциального уравнения на функции $u_1(x, y)$ в точке $x = 1/2$, $y = 1/2$ равна

$$\frac{\partial^2 u_1}{\partial x^2} + \frac{\partial^2 u_1}{\partial y^2} - (u_1 - 1)^3 = -3{,}15 \cdot 10^{-2} \cdot 2\pi^2 - (3{,}15 \cdot 10^{-2} - 1)^3 \simeq 0{,}2.$$

Чтобы повысить точность аппроксимации уравнения внутри D, следует взять больше слагаемых в сумме (11.1.7). Но с ростом m увеличивается количество гармоник $\sin p\pi x$, $\sin q\pi y$, участвующих в вычислениях интегралов, что в значительной степени затрудняет реализацию вариационного метода.

Отмеченного недостатка лишен специальный вариант метода дискретизации Ритца — метод конечных элементов.

Методом конечных элементов называют метод Ритца с выбором базисных кусочно-полиномиальных функций $e_1(x, y)$, отличных от нуля, в малой окрестности некоторых точек области D.

Точность приближения в методе конечных элементов повышается не за счет увеличения числа различных видов базисных функций (в примере — различных гармоник), а за счет уменьшения размера окрестности, где базисная функция отлична от нуля. Сам же вид базисной функции сохраняется. Здесь имеется полная аналогия со сплайн-аппроксимацией функции (см. гл. 6). С методом конечных элементов можно познакомиться в [18].

В методе дискретизации Ритца существенную роль играет выбор базиса функций, учитывающих специфику задачи. Фактически хороший выбор связан с достаточно большой предварительной работой по учету симметрии области, построению собственных функций дифференциального оператора и т. д. Естественно, что такая работа затем должна окупиться на серийных вычислениях.

● 11.2. Разностный метод решения стационарных уравнений

11.2.1. Стационарные уравнения. Уравнения с частными производными для функций $u(x, y, z)$, зависящих только от пространственных координат, называются *стационарными уравнениями.* Такие уравнения могут иметь естественное происхождение и описывать не изменяющиеся во времени процессы различной природы. Примеры стационарных процессов и уравнений приведены в п. 5.3.19. Многие стационарные линейные уравнения имеют вид

$$\operatorname{div}\bigl(p(x, q, z)\operatorname{grad}u(x, y, z)\bigr)+q(x, y, z)u=f(x, y, z).$$

Это уравнение при p, $q=\text{const}$ является уравнением Пуассона

$$\Delta u=f,$$

а при $f=0$ — уравнением Лапласа

$$\Delta u=0.$$

К стационарным уравнениям приходят и при решении нестационарных уравнений. Пусть например, в волновом уравнении

$$\frac{\partial^2 u}{\partial t^2}-a^2\Delta u=f$$

внешнее возмущение $f(x, y, z, t)$ представляется в виде

$$f=f_0(x, y, z)e^{i\omega t}.$$

Если искать решение $u(x, y, z, t)$ с той же частотой и неизвестной амплитудой $u_0(x, y, z)$

$$u = u_0(x, y, z)\, e^{i\omega t},$$

то для функции $u_0(x, y, z)$ получим стационарное уравнение

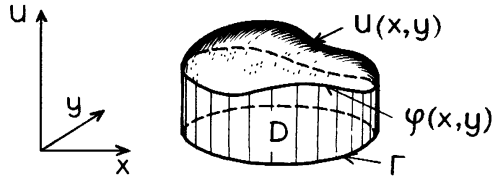

Рис. 11.4

$$\Delta u_0 + \frac{\omega^2}{a^2} u_0 = -\frac{f_0}{a^2}.$$

Это так называемое уравнение Гельмгольца.

11.2.2. Дискретизация стационарных уравнений конечно разностным методом. Идея дискретизации задач для уравнений с частными производными с помощью разностных схем (или методом сеток) та же, что и в обыкновенных дифференциальных уравнениях (см. гл. 10). Продемонстрируем ее на примере решения задачи Дирихле для уравнения эллиптического типа

$$a(x, y)\frac{\partial^2 u}{\partial x^2} + b(x, y)\frac{\partial^2 u}{\partial y^2} + c(x, y)\frac{\partial u}{\partial x} +$$

$$+ d(x, y)\frac{\partial u}{\partial y} + g(x, y)u = f(x, y), \qquad (11.2.1)$$

заданного в односвязной области D с границей Γ (рис. 11.4). Коэффициенты в уравнении, правая часть и граница Γ достаточно гладкие: $a(x, y) > 0$, $b(x, y) > 0$, $g(x, y) \leqslant 0$ в D. Пусть для уравнения (11.2.1) ставится краевая задача Дирихле

$$u(x, y) = \varphi(x, y), \quad (x, y) \in \Gamma, \qquad (11.2.2)$$

где $\varphi(x, y)$ — гладкая функция на Γ.

Покроем область D плоскости (x, y) сеткой параллельных прямых (рис. 11.5)

$$\begin{aligned} x_i &= x_0 + ih, \\ y_k &= y_0 + kh, \end{aligned} \quad i, k = 0, \pm 1, \pm 2, \ldots.$$

Точки пересечения этих прямых называются узлами. Будем рассматривать только узлы, расположенные внутри области D. Два узла (x_i, y_k) называются *соседними*, если расстояние между ними по оси x или по оси y равно h.

Узлы, которые имеют всех четырех соседей, расположенных внутри области D, называются *внутренними*. Множество всех внутренних узлов называется сеточной областью D_h (на рис. 11.5 они обозначены кружком).

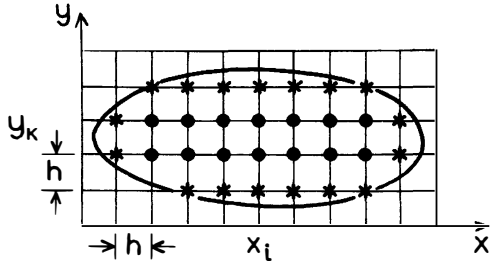

Рис. 11.5

Узлы, у которых хотя бы один соседний узел не принадлежит D, называются *граничными*. Множество всех граничных узлов называется *границей сеточной области* Γ_h (на рис. 11.5 они обозначены звездочкой).

При сделанных выше предположениях существует точное решение краевой задачи (11.2.1), (11.2.2). Это функция $u(x, y)$, а если коэффициенты, правая часть, граница и $\varphi(x, y)$ достаточно гладкие, то можно предположить, что $u(x, y)$ имеет непрерывные производные по x, y до 4-го порядка включительно, т. е.

$$u(x, y) \in C^4 [D].$$

Заменим во внутренних узлах производные в (11.2.1) разностными отношениями второго порядка точности аппроксимации по формулам

$$\frac{\partial u}{\partial x}(x_i, y_k) = \frac{u(x_{i+1}, y_k) - u(x_{i-1}, y_k)}{2h} + O(h)^2,$$

$$\frac{\partial u}{\partial x}(x_i, y_k) = \frac{u(x_i, y_{k+1}) - u(x_i, y_{k-1})}{2h} + O(h)^2,$$

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_k) = \frac{u(x_{i+1}, y_k) - 2u(x_i, y_k) + u(x_{i-1}, y_k)}{h^2} + O(h)^2,$$

$$\frac{\partial^2 u}{\partial y^2}(x_i, y_k) = \frac{u(x_i, y_{k+1}) - 2u(x_i, y_k) + u(x_i, y_{k-1})}{h^2} + O(h)^2.$$

Подставляя эти соотношения в (11.2.1), отбросив погрешность аппроксимации производных, получим разностные уравнения для неизвестных $u_{i,k}$

$$a_{i,k}\frac{u_{i+1,k} - 2u_{i,k} + u_{i-1,k}}{h^2} + b_{i,k}\frac{u_{i,k+1} - 2u_{i,k} + u_{i,k-1}}{h^2} +$$

$$+ c_{i,k}\frac{u_{i+1,k} - u_{i-1,k}}{2h} + d_{i,k}\frac{u_{i,k-1} - u_{i,k-1}}{2h} + g_{i,k}u_{i,k} = f_{i,k}, \qquad (11.2.3)$$

где введены следующие обозначения значений коэффициентов и правой части в узле (x_i, y_k): $a_{i,k}$, $b_{i,k}$, $c_{i,k}$, $d_{i,k}$, $g_{i,k}$, $f_{i,k}$, например

$$f_{i,k} = f(x_i, y_k), \quad (x_i, y_k) \in D_h.$$

Соотношения (11.2.3) содержат кроме неизвестных $u_{i,k}$ во внутренних узлах еще и неизвестные $u_{i,k}$ на границе сеточной области. Для граничных узлов запишем соотношение

$$u(x_i, y_k) = \frac{\theta u(x_{i+1}, y_k) + \varphi(x_i \pm \theta h, y_k)}{\theta + 1} + O(h)^2$$

либо

$$u(x_i, y_k) = \frac{\theta u(x_i, y_{k+1}) + \varphi(x_i, y_k \pm \theta h)}{\theta + 1} + O(h)^2$$

в зависимости от того, какая точка, $(x_i \pm \theta h, y_k)$ или $(x_i, y_k \pm \theta h)$, $0 \leqslant \theta < 1$, пересечения непрерывной границы Γ с линиями сетки

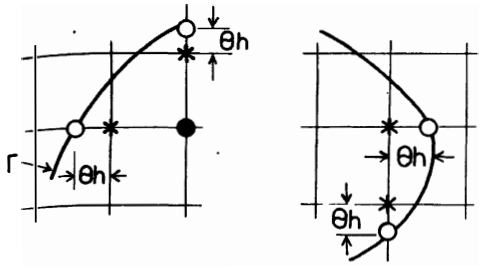

Рис. 11.6

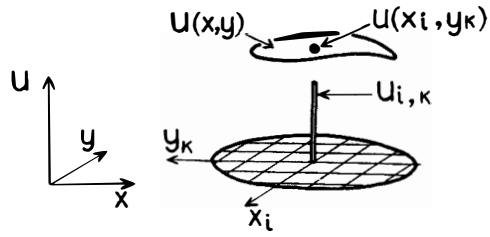

Рис. 11.7

находится ближе к граничному узлу (рис. 11.6). Эти соотношения означают, что значение $u(x_i, y_k)$ при $(x_i, y_k) \in \Gamma_h$ получается линейной интерполяцией значений $u(x, y)$ во внутреннем узле и в точке пересечения Γ с сеткой. Отбросив в последних соотношениях погрешность аппроксимации, получим выражения для неизвестных $u_{i,k}$ в граничных узлах

$$u_{i,k} = \frac{1}{(\theta+1)}(\theta u_{i\pm1,k} + \varphi_{i\pm\theta,k}),$$

или (11.2.4)

$$u_{i,k} = \frac{1}{(\theta+1)}(\theta u_{i,k\pm1} + \varphi_{i,k\pm\theta}),$$

где введено обозначение $\varphi_{i\pm\theta,k} = \varphi(x_i \pm \theta h,\ y_k)$ аналогично $\varphi_{i,k\pm\theta}$. Заметим, что в (11.2.4) дробная часть шага h (величина θ) зависит от узла $\theta = \theta_{i,k}$. Чтобы не усложнять запись в (11.2.4), индексы i и k опущены.

Присоединяя уравнения (11.2.4) к (11.2.3), получаем систему линейных алгебраических уравнений относительно $u_{i,k}$. В этой системе число уравнений равно числу неизвестных и равно числу узлов в D_h и Γ_h.

Система уравнений (11.2.3), (11.2.4) — это разностная схема непрерывной задачи (11.2.1), (11.2.2).

Решения разностной схемы $u_{i,k}$ — приближения к точному решению $u(x_i, y_k)$ в узлах x_i, y_k (рис. 11.7). Перепишем систему уравнений (11.2.3) в следующем виде:

$$L_h u_{i,k} = f_{i,k}, \qquad (11.2.5)$$

$$L_h u_{i,k} = A_{i,k} u_{i,k-1} + B_{i,k} u_{i-1,k} + C_{i,k} u_{i,k} + D_{i,k} u_{i+1,k} + E_{i,k} u_{i,k+1},$$

где коэффициенты задаются формулами

$$A_{i,k} = \frac{b_{i,k}}{h^2} - \frac{d_{i,k}}{2h},\ B_{i,k} = \frac{a_{i,k}}{h^2} - \frac{c_{i,k}}{2h},$$

$$C_{i,k} = -\frac{2(a_{i,k}+b_{i,k})}{h^2} + g_{i,k},$$

$$D_{i,k} = \frac{a_{i,k}}{h^2} + \frac{c_{i,k}}{2h},\ E_{i,k} = \frac{b_{i,k}}{h^2} + \frac{d_{i,k}}{2h}.$$

По предположению, гладкие функции $a(x, y) > 0$, $b(x, y) < 0$, $g(x, y) \leqslant 0$, поэтому

$$g_{i,k} \leqslant 0,\ A_{i,k} > 0,\ B_{i,k} > 0,\ C_{i,k} < 0,\ D_{i,k} > 0,\ E_{i,k} > 0 \quad (11.2.6)$$

для достаточно малого шага h. Заметим, что имеет место равенство

$$A_{i,k} + B_{i,k} + C_{i,k} + D_{i,k} + E_{i,k} = g_{i,k}. \quad (11.2.7)$$

Будем предполагать выбор h столь малым, что сеточная область D_h остается односвязной, т. е. из любого узла D_h в любой другой можно перейти на сетке по пути, который связывает только соседние узлы. Важным фактом разностных уравнений (11.2.5) является следующая теорема, которая называется *принципом максимума*. На принципе максимума основано доказательство многих теорем существования и единственности теории разностных схем.

Теорема 11.3. *Пусть для значений $u_{i,k}$ на множестве внутренних узлов выполняется неравенство*

$$L_h u_{i,k} \geqslant 0;$$

тогда на D_h значения $u_{i,k}$ не могут иметь положительного максимума; если

$$L_h u_{i,k} \leqslant 0,$$

то на D_h значения $u_{i,k}$ не могут иметь отрицательного минимума, за исключением случая $u_{i,k} = \text{const}$.

Доказательство. Допустим, что $u_{i,k} \neq \text{const}$ и во всех внутренних узлах $L_h u_{i,k} \geqslant 0$. Предположим противное, т. е. что $u_{i,k}$ достигает положительного максимума $U > 0$ в некотором внутреннем узле. Тогда можно найти такой внутренний узел (i_*, k_*), в котором $u_{i_*,k_*} = U > 0$ и хотя бы в одном соседнем с ним узле $u_{i,k} < U$ (это следует из допущения $u_{i,k} \neq \text{const}$). Запишем

$$L_h u_{i_*,k_*} = A_{i_*,k_*} u_{i_*,k_*+1} + B_{i_*,k_*} u_{i_*-1,k_*} + C_{i_*,k_*} U + D_{i_*,k_*} u_{i_*+1,k_*} + E_{i_*,k_*} u_{i_*,k_*-1} \geqslant 0.$$

Из неравенств (11.2.6) следует, что замена значений $u_{i,k}$ в соседних с i_*, k_* узлах на U приведет к строгому неравенству

$$U(A_{i_*,k_*} + B_{i_*,k_*} + C_{i_*,k_*} + D_{i_*,k_*} + E_{i_*,k_*}) > 0,$$

но из (11.2.7) следует

$$U g_{i_*,k_*} > 0,$$

что противоречит условию $g_{i,k} < 0$ в D_h. Следовательно, сделанное предположение неверно и первая часть теоремы доказана, вторая часть доказывается аналогично.

Упростим выражения в граничных узлах (11.2.4), полагая $\theta = 0$, что соответствует переносу значения $\varphi(x, y)$ с непрерывной границы в узел Γ_h. Погрешность такой аппроксимации $O(h)$.

Теорема 11.4. *Пусть $\theta = 0$, тогда система линейных уравнений* (11.2.5), (11.2.4) *имеет единственное решение.*

Доказательство. Теорема будет доказана, если показать, что соответствующая система (11.2.5), (11.2.4) — однородная линей-

ная система — имеет только нулевое решение ($u_{i,k}=0$). Однородная система получается из (11.2.5), (11.2.4), если положить

$$f_{i,k}=0,\quad \varphi_{i,k}=0.$$

Таким образом, получаем значения $u_{i,k}=0$ для граничных узлов из условия $u_{i,k}=\varphi_{i,k}$. Пусть хотя бы в одном внутреннем узле $u_{i_*,k_*}>0$ (<0), но тогда по принципу максимума (теорема 11.3) максимальное положительное (или отрицательное минимальное) значение $u_{i,k}$ достигается на границе Γ_h, что невозможно, так как $u_{i,k}=0$ на границе Γ_h. Противоречие доказывает теорему.

В общем случае, когда в уравнениях (11.2.4) $\theta\neq 0$, также справедливо аналогичное утверждение.

Теорема 11.5. *Система линейных уравнений* (11.2.5), (11.2.4) *имеет единственное решение.*

Доказательство. Покажем, что однородная система (11.2.5), (11.2.4) имеет только нулевое решение. Если хотя бы в одном внутреннем узле $u_{i_*,k_*}>0$ (<0), то, согласно принципу максимума, максимальное положительное (минимальное отрицательное) значение $u_{i,k}$ достигается на границе, что противоречит (11.2.4), поскольку на границе Γ_h имеем

$$\left|u_{i,k}\right|=\frac{\theta}{1+\theta}\left|u_{i\pm 1,k\pm 1}\right|.$$

Полученное противоречивое неравенство $\dfrac{\theta}{1+\theta}>1$ и доказывает теорему.

11.2.3. Решение разностных уравнений. Для решения системы линейных уравнений полученной разностной схемы могут применяться методы, изложенные в гл. 8, например метод простой итерации. Перепишем систему уравнений (11.2.5), (11.2.4) в виде, удобном для применения метода простой итерации: для внутренних узлов

$$u_{i,k}=-\frac{A_{i,k}}{C_{i,k}}u_{i,k+1}-\frac{B_{i,k}}{C_{i,k}}u_{i-1,k}-\frac{D_{i,k}}{C_{i,k}}u_{i+1,k}-\frac{E_{i,k}}{C_{i,k}}u_{i,k+1}+\frac{f_{i,k}}{C_{i,k}},\tag{11.2.8}$$

для граничных узлов

$$u_{i,k}=\frac{\theta}{1+\theta}u_{i\pm 1,k\pm 1}+\frac{1}{1+\theta}\varphi_{i\pm\theta,k\pm\theta}.\tag{11.2.9}$$

Предположим, что $g_{i,k}<0$ и выполняются условия (11.2.6). Будем решать систему уравнений относительно $u_{i,k}$ методом простой итерации согласно следующему итерационному процессу: для внутренних узлов

$$u_{i,k}^{(p+1)}=-\frac{A_{i,k}}{C_{i,k}}u_{i,k+1}^{(p)}-\frac{B_{i,k}}{C_{i,k}}u_{i-1,k}^{(p)}-\frac{D_{i,k}}{C_{i,k}}u_{i+1,k}^{(p)}-\frac{E_{i,k}}{C_{i,k}}u_{i,k+1}^{(p)}+\frac{f_{i,k}}{C_{i,k}},$$

для граничных узлов

$$u_{i,k}^{(p+1)}=\frac{\theta}{1+\theta}u_{i\pm1,k\pm1}^{(p)}+\frac{1}{1+\theta}\varphi_{i\pm\theta,k\pm\theta};\quad p=0,\ 1,\ 2,\ \ldots$$

$u_{i,k}^{(0)}$ задано.

Теорема 11.6. *Пусть* $g_{i,k}<0$ *и выполнены условия* (11.2.6). *Тогда последовательные приближения* $u_{i,k}^{(p)}$ *сходятся к точному решению разностной схемы* $u_{i,k}$ *или системы уравнений* (11.2.8), (11.2.9) *и имеет место оценка*

$$\max_{i,k}|u_{i,k}^{(p)}-u_{i,k}|\leqslant\frac{q^p}{1-q}\max_{i,k}|u_{i,k}^{(0)}|,$$

где

$$q=\max_{i,k}\left(\frac{\theta_{i,k}}{1+\theta_{i,k}},-\frac{A_{i,k}+B_{i,k}+D_{i,k}+E_{i,k}}{C_{i,k}}\right).$$

Доказательство этой теоремы состоит в проверке условия сходимости метода простой итерации для систем линейных уравнений (см. гл. 6), при этом подразумевается, что неизвестный вектор v образуют элементы $u_{i,k}$. Например, компоненты вектора v можно занумеровать следующим образом: пусть $1\leqslant i\leqslant N_1$, $1\leqslant k\leqslant N_2$; тогда

$$v_1=u_{1,1},\ v_2=u_{2,1},\ \ldots,\ v_{N_1}=u_{N_1,1},$$
$$v_{N_1+1}=u_{2,1},\ v_{N_1+2}=u_{2,2},\ \ldots,\ v_{2N_1}=u_{N_1,2},$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ v_{N_1,N_2}=u_{N_1,N_2}.$$

Относительно вектора v разностная схема — это система линейных уравнений в матричной записи

$$Av=b,$$

где матрица A имеет в каждой строке не более пяти ненулевых элементов:

$$A=\begin{pmatrix}
**\cdots\cdots\cdots\cdots\cdots\\
***\cdots\cdots\cdots\cdots\cdots\\
\cdots\cdots\cdots\cdots\cdots\cdots\\
\cdots*\cdots***\cdots*\cdots\cdots\\
\cdots*\cdots***\cdots*\cdots\cdots\\
\cdots\cdots\cdots\cdots\cdots\cdots\\
\cdots\cdots\cdots\cdots\cdots**
\end{pmatrix}\qquad(11.2.10)$$

Это связано с тем, что производные в каждом внутреннем узле $(i,\ k)$ аппроксимировались по пяти соседним узлам (рис. 11.8).

Узлы в аппроксимации производных дифференциального уравнения называются *шаблоном*.

На рис. 11.8 изображен пятиточечный шаблон, который использовался при построении разностной схемы во внутренних узлах исходного дифференциального уравнения.

Матрица A вида (11.2.10), у которой достаточно много нулевых элементов, называется *разреженной*. Для решения систем линейных уравнений с такими матрицами разработаны достаточно эффективные прямые методы решения — формы исключения Гаусса со специальным хранением матрицы (упаковкой) в памяти ЭВМ и специальной обработкой. Подробности можно найти в [25].

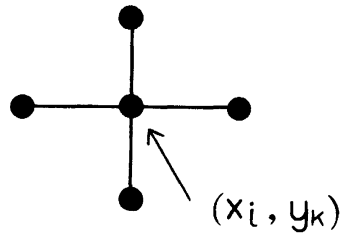

Рис. 11.8

11.2.4. Оценка погрешности и сходимость решений разностных уравнений. Можно показать, что принцип максимума, справедливый для системы разностных уравнений, эквивалентен устойчивости разностной схемы. Но тогда, так же как и для обыкновенных дифференциальных уравнений, решения разностных уравнений при $h\to 0$ сходятся к точному решению краевой задачи со скоростью, определяемой порядком аппроксимации уравнения и краевых условий. Таким образом, для точного решения $(u(x,\ y)\in C^4[D])$ имеем оценку погрешности

$$\max_{i,\,k}|u_{i,\,k}-u(x_i,\ y_k)|=O(h^2),\quad h\to 0. \tag{11.2.11}$$

Оценка погрешности (11.2.11) справедлива, если точное решение четырежды непрерывно дифференцируемо в области D. Для областей с угловыми точками, например прямоугольника, вообще говоря, $u(x,\ y)\notin C^4[D]$. Однако если граничная функция, т. е. $\varphi(x,\ y)$, удовлетворяет в углах специальным условиям согласования, то точное решение $u(x,\ y)\in C^4[D]$ и является верной оценка (11.2.11).

Для прямоугольной области $D=\{x_0\leqslant x\leqslant x_1,\ y_0\leqslant y\leqslant y_1\}$ такими условиями согласования могут быть:

1) достаточная гладкость $\varphi(x,\ y)$,

2) функция $\varphi(x,\ y)$ должна удовлетворять в углах прямоугольника дифференциальному уравнению.

Например, функция $\varphi(x,\ y)=x^2+y^2$ удовлетворяет условиям согласования для уравнения

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=4$$

в углах D.

Оценка погрешности (11.2.11) имеет в основном теоретическое значение, поскольку содержит константу c, которую практически трудно определить:

$$\max_{i,\,k}|u_{i,\,k}-u(x_i,\ y_k)|=ch^2+o(h^2),\quad h\to 0.$$

Поэтому в реальных расчетах используется правило Рунге оценки погрешности, аналогичное тому, которое применяется в численном интегрировании и решении обыкновенных дифференциальных

уравнений. Проводятся два варианта расчета $u^h_{i,k}$ с шагом h и $u^{h/2}_{i,k}$ с шагом $h/2$, тогда погрешность имеет вид

$$\max_{i,k}|u^{h/2}_{i,k}-u(x_i,\ y_i)|=\frac{1}{3}\max_{i,k}|u^{h/2}_{i,k}-u^h_{i,k}|+o(h^2)$$

и главная часть погрешности определяется на совпадающих узлах.

11.2.5. Пример построения разностной схемы. Для примера рассмотрим уравнение

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+e^{-x}\frac{\partial u}{\partial x}-u=f(x,\ y) \tag{11.2.12}$$

в квадрате $D=\{0\leqslant x\leqslant 1,\ 0\leqslant y\leqslant 1\}$ с краевым условием

$$u(x,\ y)=\varphi(x,\ y)=\exp(x-y) \tag{11.2.13}$$

на сторонах квадрата D.

Чтобы функция $\varphi(x,\ y)$ удовлетворяла условиям согласования, подставим (11.2.13) в (11.2.12) и таким образом определим функцию $f(x,\ y)$. Имеем

$$f(x,\ y)=e^{-y}(e^x+1). \tag{11.2.14}$$

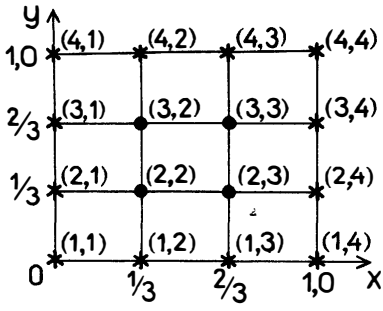

Рис. 11.9

Следовательно, точное решение уравнения (11.2.12) с краевыми условиями (11.2.13) — это и есть функция (11.2.13), определенная внутри области D.

Метод построения этого примера с известным точным решением краевой задачи является традиционным при отладке программ решения рассматриваемого класса задач.

Выберем шаг $h=1/3$. Занумеруем узлы так, как показано на рис. 11.9. Для значений $u_{i,k}$ в граничных узлах из (11.2.13) получаем

$$u_{1,1}=1{,}0,\quad u_{1,2}=\exp(1/3),\quad u_{1,3}=\exp(2/3),\quad u_{1,4}=\exp(1),$$
$$u_{2,4}=\exp(2/3),\quad u_{3,4}=\exp(1/3),\quad u_{4,4}=1{,}0,\quad u_{4,3}=\exp(-1/3),$$
$$u_{4,2}=\exp(-2/3),\quad u_{4,1}=\exp(-1),\quad u_{3,1}=\exp(-2/3),\quad u_{2,1}=\exp(-1/3).$$

Для значений во внутренних узлах из (11.2.12), (11.2.14) получаем систему четырех уравнений с четырьмя неизвестными:

$$\frac{u_{2,3}-2u_{2,2}+u_{2,1}}{(1/3)^2}+\frac{u_{3,2}-2u_{2,2}+u_{1,2}}{(1/3)^2}+e^{-1/3}\frac{(u_{2,3}-u_{2,1})}{(2/3)}-u_{2,2}=e^{-1/3}(e^{1/3}+1),$$

$$\frac{u_{2,4}-2u_{2,3}+u_{2,2}}{(1/3)^2}+\frac{u_{3,3}-2u_{2,3}+u_{1,3}}{(1/3)^2}+e^{-2/3}\frac{(u_{2,4}-u_{2,2})}{(2/3)}-u_{2,3}=e^{-2/3}(e^{2/3}+1),$$

$$\frac{u_{3,4}-2u_{3,3}+u_{3,2}}{(1/3)^2}+\frac{u_{4,3}-2u_{3,3}+u_{2,3}}{(1/3)^2}+e^{-2/3}\frac{(u_{3,4}-u_{3,2})}{(2/3)}-u_{3,3}=e^{-2/3}(e^{2/3}+1),$$

$$\frac{u_{3,3}-2u_{3,2}+u_{3,1}}{(1/3)^2}+\frac{u_{1,2}-2u_{3,2}+u_{2,2}}{(1/3)^2}+e^{-1/3}\frac{u_{2,3}-u_{2,1}}{(2/3)}-u_{3,2}=e^{-2/3}(e^{1/3}+1).$$

Решив эту систему уравнений относительно $u_{2,2}$, $u_{2,3}$, $u_{3,3}$, $u_{3,2}$, определим приближенные значения к соответствующим точным значениям решения $u(1/3,\ 1/3)$, $u(2/3,\ 1/3)$, $u(2/3,\ 2/3)$, $u(1/3,\ 2/3)$.

11.2.6. Применение программы B8A0. Программа решает разностные уравнения, которые получаются при аппроксимации дифференциального уравнения на пятиточечном шаблоне. Область решения краевой задачи — прямоугольник. Если задана другая область, то ее можно заключить в прямоугольник и на непересекающейся части области с прямоугольником задать специальным образом краевую задачу. Затем используется программа B8A0. Более подробно с изложенным приемом, который называется методом фиктивных областей, можно познакомиться в [16].

Напишем программу решения задачи (11.2.12), (11.2.13) на сетке с числом узлов 8×8, т. е. с шагом $h=1/7$. Приведем формулы для вычисления коэффициентов разностной схемы (11.2.5) для внутренних узлов $2\leqslant i\leqslant 7,\ 2\leqslant k\leqslant 7$:

$$A_{i,k}=\frac{1}{h^2};\ B_{i,k}=\frac{1}{h^2}-\frac{e^{-x_i}}{2h};\ C_{i,k}=-\frac{4}{h^2}-1;$$

$$D_{i,k}=\frac{1}{h^2}+\frac{e^{-x_i}}{2h};\ E_{i,k}=\frac{1}{h^2}.$$

Здесь $x_i=(i-1)h;\ y_k=(k-1)h$.

Для внешних узлов $1\leqslant i,\ k\leqslant 8$

$$A_{i,1}=0,\ A_{i,8}=0,\ A_{1,k}=0,\ A_{8,k}=0,$$
$$B_{i,1}=0,\ B_{i,8}=0,\ B_{1,k}=0,\ B_{8,k}=0,$$
$$C_{i,1}=1,\ C_{i,8}=1,\ C_{1,k}=1,\ C_{8,k}=1,$$
$$D_{i,1}=0,\ D_{i,8}=0,\ D_{1,k}=0,\ D_{8,k}=0,$$
$$E_{i,1}=0,\ E_{i,8}=0,\ E_{1,k}=0,\ E_{8,k}=0.$$

Формулы $f_{i,k}$ для внутренних узлов

$$f_{i,k}=e^{-y_k}(e^{x_i}+1),\ 2\leqslant i,\ k\leqslant 7.$$

Формулы $f_{i,k}$ для внешних узлов

$$f_{i,1}=e^{x_i-y_1},\ f_{i,8}=e^{x_i-y_8},\ f_{1,k}=e^{x_1-y_k},\ f_{8,k}=e^{x_8-y_k},\ 1\leqslant i,\ k\leqslant 8.$$

Зададим точность выполнения разностных уравнений $\varepsilon_1=10^{-3}$, точность для прекращения итераций $\varepsilon_2=10^{-4}$.

Программа может иметь следующий вид:

```
      REAL  H,X(8),Y(8),A(8,8),B(8,8),C(8,8),D(8,8)
      REAL  E(8,8),F(8,8),U(8,8),G,E1,E2
      REAL  E3(20),E4(20),W1(8,8),W2(8,8),W3(8,8)
      INTEGER  N1,N2,N,I1,I2,I3,N3,I4,I5,I
      DATA  H/0.142857/,N1,N2,N,I1,I2,N3,I4,I5,I
     * /8,8,8,20,0,0,0,0,0/
```

```
      DATA E1,E2/1.E-3,1.E-4/
C     ВЫЧИСЛЕНИЕ ЗНАЧЕНИЙ УЗЛОВ И КОЭФФИЦИЕН-
C     ТОВ РАЗНОСТНОГО УРАВНЕНИЯ
      DO 1 I=1,8
      X(I)=(I-1)*H
1     Y(I)=(I-1)*H
      DO 2 I=2,7
      DO 2 K=2,7
      A(I,K)=1./(H*H)
      B(I,K)=1./(H*H)-(EXP(-X(I))/2.*H)
      C(I,K)=-4./(H*H)-1.
      D(I,K)=1./(H*H)+(EXP(-X(I))/2.*H)
      E(I,K)=1./(H*H)
2     F(I,K)=EXP(-Y(K))*(EXP(X(I))+1.)
      DO 3 I=1,8
      A(I,1)=0.
      B(I,1)=0.
      C(I,1)=1.
      D(I,1)=0.
      E(I,1)=0.
      A(I,8)=0.
      B(I,8)=0.
      C(I,8)=1.
      D(I,8)=0.
      E(I,8)=0.
      A(1,I)=0.
      B(1,I)=0.
      C(1,I)=1.
      D(1,I)=0.
      E(1,I)=0.
      A(8,I)=0.
      B(8,I)=0.
      C(8,I)=1.
      D(8,I)=0.
      E(8,I)=0.
      F(I,1)=EXP(X(I)-Y(1))
      F(I,8)=EXP(X(I)-Y(8))
      F(1,I)=EXP(X(1)-Y(I))
3     F(8,I)=EXP(X(8)-Y(I))
C     ОБРАЩЕНИЕ К ПРОГРАММЕ B8A0
      CALL B8A0(N1,N2,N,A,B,C,D,E,F,U,G,I1,I2,I3,
     * N3,I4,I5,E1,E2,E3,E4,W1,W2,W3,I)
C     ВЫВОД НА АЦПУ РЕШЕНИЯ РАЗНОСТНЫХ УРАВ-
C     НЕНИЙ
      WRITE (6,4) U
4     FORMAT (2X,8E13.6)
      END
```

● 11.3. Разностный метод решения нестационарных уравнений

11.3.1. Нестационарные уравнения. Уравнения с частными производными, у которых одна из независимых переменных является временем, называются *нестационарными*. Решения таких уравнений $u(t, x, y, z)$ описывают изменяющиеся во времени и пространстве процессы различной природы. Примеры нестационарных уравнений (волнового и теплопроводности) приведены в п. 5.3.19.

Особая роль времени в нестационарных уравнениях обусловлена также тем, что нестационарные процессы часто бывают необратимыми во времени. Это связано с диссипацией (рассеиванием) энергии, происходящей во время процесса. В диссипативных уравнениях неэквивалентность положительного хода времени и отрицательного проявляется в том, что преобразование $t_1 = -t$ не сохраняет вида уравнений.

11.3.2. Волновое уравнение. Рассмотрим смешанную задачу (т. е. заданы начальные и краевые условия) для волнового уравнения

$$\frac{\partial^2 u}{\partial t^2} = a^2 \frac{\partial^2 u}{\partial x^2} + f(x, t) \tag{11.3.1}$$

в области $D = \{0 \leqslant x \leqslant l,\ 0 \leqslant t \leqslant t_1\}$ с начальными условиями

$$u(x, 0) = \varphi_0(x),\ \frac{\partial u}{\partial t}(x, 0) = \varphi_1(x) \tag{11.3.2}$$

и условиями на краях

$$u(0, t) = 0,\ u(l, t) = 0. \tag{11.3.3}$$

Будем предполагать, что $f(x, t)$, $\varphi_0(x)$, $\varphi_1(x)$ — достаточно гладкие функции, причем выполнены условия согласования в двух углах области D $(x=0,\ t=0)$, $(x=l,\ t=0)$, обеспечивающие существование и единственность решения $u(x, t) \in C^4[D]$.

Для дискретизации исходной задачи покроем (рис. 11.10) область сеточной областью $D_{h,\tau}$:

$$D_{h,\tau} = \{x_i = ih,\ 0 \leqslant i \leqslant m,\ t_j = j\tau,\ 0 \leqslant j \leqslant p\},$$

где $m = l/h$, $p = t_1/\tau$. Заменим производные во внутренних узлах $D_{h,\tau}$ конечно-разностными отношениями

$$\frac{\partial^2 u}{\partial t^2}(x_i, t_j) = \frac{u(t_{j+1}, x_i) - 2u(t_j, x_i) + u(t_{j-1}, x_i)}{\tau^2} + O(\tau^2),$$

$$\frac{\partial^2 u}{\partial x^2}(t_j, x_i) = \frac{u(t_j, x_{i+1}) - 2u(t_j, x_i) + u(t_j, x_{i-1})}{h^2} + O(h^3)$$

по шаблону, указанному на рис. 11.10. Подставим эти выражения в (11.3.2); отбрасывая погрешность аппроксимации, получим

$$\frac{1}{\tau^2}(u_{j+1,i} - 2u_{j,i} + u_{j-1,i}) = \frac{a^2}{h^2}(u_{j,i+1} - 2u_{j,i} + u_{j,i-1}) + f_{j,i} \tag{11.3.4}$$

— систему разностных уравнений для внутренних узлов, $1 \leqslant j \leqslant p-1$, $1 \leqslant i \leqslant m-1$. Начальные условия (11.3.2) заменяем разностными соотношениями

$$u_{i,0} = \varphi_0(x_i), \quad 0 \leqslant i \leqslant m,$$

$$u_{i,1} = \varphi_0(x_i) + \tau\varphi_1(x_i) + \frac{\tau^2}{2}\left(f_{i,0} + a^2 \frac{\varphi_0(x_{i+1}) - 2\varphi_0(x_i) + \varphi_0(x_{i-1})}{h^2}\right). \quad (11.3.5)$$

Краевые условия заменяем условиями в узлах

$$u_{0,j} = 0, \quad u_{m,j} = 0, \quad 0 \leqslant j \leqslant p. \quad (11.3.6)$$

Система линейных алгебраических уравнений относительно $u_{i,j}$ (11.3.4)—(11.3.6) является разностной схемой смешанной задачи (11.3.1)—(11.3.3). Погрешность аппроксимации замены дифференциального уравнения и начального условия $O(h^2+\tau^2)$.

Заметим, что как в стационарном уравнении, так и в волновом использовался одинаковый шаблон. Но на границах области заданы различные условия. Смешанные условия (11.3.5), (11.3.6) позволяют явно записать формулы для решений разностных уравнений на $(j+1)$-м временном слое, если известны $u_{i,j}$, $u_{i,j-1}$, т. е. явно решить систему разностных уравнений.

Схемы, позволяющие явно записать решения систем уравнений разностной схемы, по аналогии с обыкновенными дифференциальными уравнениями называются *явными*.

Из (11.3.4) для значений $u_{i,j+1}$ на $(j+1)$-м временном слое находим формулу

$$u_{i,j+1} = 2u_{i,j} - u_{i,j-1} + \frac{a^2\tau^2}{h^2}(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}) + \tau^2 f_{i,j}, \quad (11.3.7)$$

которая совместно с (11.3.5.), (11.3.6) позволяет вычислить значения решения на втором слое

$$u_{0,2};\ u_{1,2};\ \ldots;\ u_{m,2} \quad (11.3.8)$$

по значениям на нулевом и первом. Затем по формуле (11.3.7) совместно с (11.3.8), (11.3.6) можно вычислить значения решения на третьем слое $u_{0,3}$; $u_{1,3}$; ...; $u_{m,3}$ по значениям на первом и втором слоях и т. д. Процесс вычислений завершается после достижения p-го временного слоя.

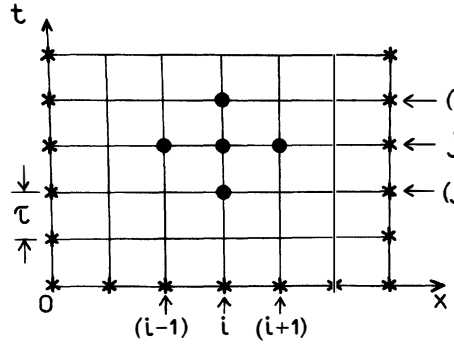

Рис. 11.10

Аналогично явным схемам для обыкновенных дифференциальных уравнений (см. **10.1**) схема (11.3.7) условно устойчива. Поэтому описанный выше процесс следует применять с учетом условий устойчивости.

Формулы явных разностных схем легко программируются. Некоторые временные слои обычно выдают на терминал, АЦПУ или графопостроитель, например, третий слой в виде, представленном на рис. 11.11.

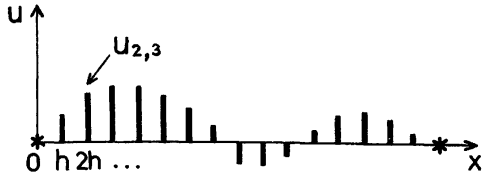

Рис. 11.11

Справедливо следующее утверждение.

Теорема 11.7. *Пусть* $u(x, t)\in C^4[D]$ *— решение задачи* (11.3.1)—(11.3.3). *Пусть также выполнено условие устойчивости разностной схемы*

$$\frac{a^2\tau^2}{h^2}<1.$$

Тогда решения разностной схемы $u_{i,j}$ при $h, \tau\to 0$ *сходятся к точному решению в среднеквадратичной норме на временном слое и имеет место оценка*

$$\max_{0\leqslant j\leqslant p}\left(h\sum_{i=0}^{m-1}(u_{i,j}-u(x_i, t_j))^2\right)^{1/2}=O(h^2+\tau^2).$$

11.3.3. Одномерное уравнение теплопроводности. Рассмотрим одномерное (одна пространственная переменная x) уравнение теплопроводности

$$\frac{\partial u}{\partial t}=a^2\frac{\partial^2 u}{\partial x^2}+f(x, t) \qquad (11.3.9)$$

в области $D=\{0\leqslant x\leqslant l,\ 0\leqslant t\leqslant t_1\}$ со следующими смешанными условиями: начальное условие

$$u(x, 0)=\varphi_0(x) \qquad (11.3.10)$$

и краевые условия

$$u(0, t)=\mu_0(t),\ \ u(l, t)=\mu_1(t). \qquad (11.3.11)$$

Будем предполагать, что $f(x, t)$ и условия (11.3.10), (11.3.11) согласованы в двух углах прямоугольника так, что $u(x, t)\in$ $\in C^4[D]$ — точное решение задачи (11.3.9)—(11.3.11). Проведем дискретизацию задачи на сеточной области $D_{h,\tau}$, той же, что и для волнового уравнения на трех шаблонах: 1) явном, 2) неявном, 3) неявном шеститочечном (рис. 11.12). Начальное условие (11.3.10) в дискретном виде запишется следующим образом (для трех шаблонов):

$$u_{i,0}=\varphi_0(x_i),$$
$$0\leqslant i\leqslant m. \qquad (11.3.12)$$

Краевые условия

$$u_{0,j}=\mu_0(t_j),\ \ u_{m,j}=\mu_1(t_j),$$
$$0\leqslant j\leqslant p. \qquad (11.3.13)$$

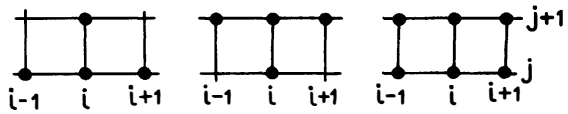

Рис. 11.12

Для внутренних узлов в соответствии с тремя шаблонами заменим в (11.3.9) производные следующими разностными отношениями:

$$1)\ \frac{1}{\tau}(u_{i,j+1}-u_{i,j})=\frac{a^2}{h^2}(u_{i+1,j}-2u_{i,j}+u_{i-1,j})+f_{i,j}; \tag{11.3.14}$$

$$2)\ \frac{1}{\tau}(u_{i,j+1}-u_{i,j})=\frac{a^2}{h^2}(u_{i+1,j+1}-2u_{i,j+1}+ +u_{i-1,j+1})+f_{i,j}; \tag{11.3.15}$$

$$3)\ \frac{1}{\tau}(u_{i,j+1}-u_{i,j})=\frac{a^2}{2h^2}[(u_{i+1,j}-2u_{i,j}+u_{i-1,j})+ +(u_{i+1,j+1}-2u_{i,j+1}+u_{i-1,j+1})]+f_{i,j}. \tag{11.3.16}$$

Заметим, что правая часть (11.3.16) является полусуммой правых частей для явной и неявной четырехточечных схем. Нетрудно проверить, что если $u(x, t)\in C^4[D]$, то на $u(x, t)$ погрешность аппроксимации для схем 1), 2) в узлах x_i, t_j такова: $O(\tau+h^2)$; для схемы 3) $O(\tau^2+h^2)$.

Разностные схемы 1), 2), 3) исходной смешанной задачи для уравнения теплопроводности — это выражения (11.3.12), (11.3.13) совместно с разностными уравнениями для внутренних узлов 1), 2), 3).

Разностная схема 1) устойчива, если выполнено неравенство $\tau a^2 \leqslant \frac{1}{2}h^2$.

Разностная схема 2) безусловно устойчива, но известный вычислительный принцип: погрешности должны быть одного порядка малости при вкладе в полную ошибку (см. гл. 5) — также требует, чтобы $\tau=O(h^2)$. Указанные жесткие ограничения на выбор шага τ по времени делают схемы 1), 2) неэффективными для практики. Например, при абсолютной точности $\varepsilon\cong 10^{-2}$ следует выбрать $h\cong 0{,}1$, $\tau\cong 0{,}01$, совершив примерно в 10 раз больше вычислительной работы, чем по схеме 3) на одном и том же временном интервале.

Наиболее употребительна для рассматриваемой задачи схема 3), которую называют *схемой Кранка — Николсона*. Эта схема лишена отмеченных двух недостатков четырехточечных схем.

Рассмотрим ход вычислительного процесса по схеме 3) на примере перехода с нулевого временного слоя на первый. На нулевом временном слое $u_{i,0}$ заданы (рис. 11.13). Чтобы найти $u_{i,1}$, $0\leqslant i\leqslant m$, необходимо решить систему линейных уравнений, которая получается из (11.3.16) при $j=0$, а именно:

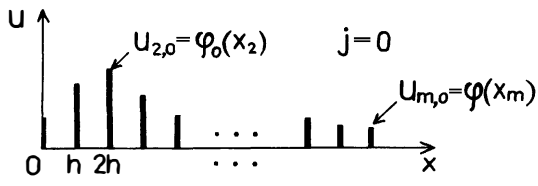

Рис. 11.13

$$\frac{1}{\tau}(u_{i,1}-u_{i,0})=\frac{a^2}{2h^2}\left[(u_{i+1,0}-2u_{i,0}+u_{i-1,0})+\right.$$
$$\left.+(u_{i+1,1}-2u_{i,1}+u_{i-1,1})\right]+f_{i,0};$$

имеем еще два соотношения для краевых значений из (11.3.13):

$$u_{0,1}=\mu_0(\tau),\quad u_{m,1}=\mu_1(\tau). \tag{11.3.17}$$

Перепишем систему уравнений для значений $u_{i,1}$ во внутренних узлах $(1\leqslant i\leqslant m-1)$ в следующем виде:

$$\frac{\tau a^2}{2h^2}u_{i-1,1}-\left(\frac{\tau a^2}{h^2}+1\right)u_{i,1}-\frac{\tau a^2}{2h^2}u_{i-1,1}=F_{i,0}, \tag{11.3.18}$$

где

$$F_{i,0}=-\tau f_{i,0}-\frac{\tau a^2}{2h^2}(u_{i+1,0}-2u_{i,0}+u_{i-1,0})-u_{i,0}$$

— известные величины, вычисляемые по нулевому слою. Теперь ясно, что (11.3.18), (11.3.17) — это трехдиагональная система линейных уравнений, которую можно решать методом прогонки (условия устойчивости прогонки здесь выполнены). Решив методом прогонки (11.3.18), (11.3.17), найдем значения $u_{i,1}$, $0\leqslant i\leqslant m$. Чтобы найти решение разностной схемы на втором временном слое $u_{i,2}$, заменим (11.3.17) из (11.3.13): $u_{0,2}=\mu_0(2\tau)$, $u_{m,1}=\mu_1(2\tau)$, а в (11.3.18) индекс 1 заменим на 2, 0 — на 1. Решив полученную трехдиагональную систему относительно $u_{i,2}$, определим значения на втором временном слое и т. д. Вычислительный процесс продолжаем до достижения p-го временного слоя.

Сходимость решений разностных уравнений к точному решению $u(x, t)$ устанавливает следующее утверждение.

Теорема 11.8. *Пусть* $u(x, t)\in C^4[D]$ *— решение задачи* (11.3.9) — (11.3.11). *Тогда решение разностной схемы* 3) — $u_{i,j}$ *при* $h,\ \tau\to 0$ *сходится к точному решению и имеет место оценка*

$$\max_{0\leqslant j\leqslant p}\ \max_{0\leqslant i\leqslant m}|u_{i,j}-u(x_i, t_j)|=O(h^2+\tau^2).$$

11.3.4. Двумерное уравнение теплопроводности. Уравнение теплопроводности с двумя пространственными координатами лежит в основе моделирования многих реальных задач тепло- и массопереноса. Вычислительные затраты решения таких задач выше, чем одномерных. Но если задача существенно двумерна, то одномерное уравнение уже нельзя использовать при построении модели.

Пусть, например, требуется найти распределение температуры в плоской квадратной пластине, на границе которой поддерживается заданная температура, с плотностью распределения на пластине тепловых источников $f(x, y, t)$. Начальное распределение температуры задано. Тогда эта задача в линейном приближении может быть записана в виде

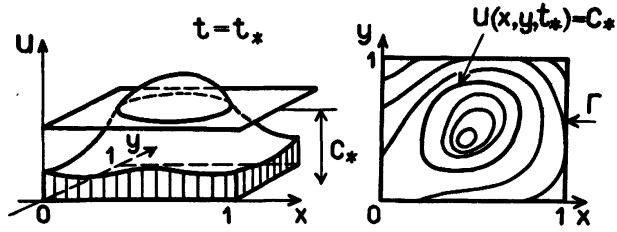

Рис. 11.14

$$\frac{\partial u}{\partial t}=a^2\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)+f(x,\,y,\,t) \qquad (11.3.19)$$

в области $D=\{0\leqslant x\leqslant 1,\ 0\leqslant y\leqslant 1,\ 0\leqslant t\leqslant t_1\}$ с начальным условием

$$u(x,\,y,\,0)=\varphi_0(x,\,y) \qquad (11.3.20)$$

и краевыми условиями

$$u(x,\,y,\,t)=\mu(x,\,y,\,t),\ \ (x,\,y)\in\Gamma. \qquad (11.3.21)$$

Решением задачи (11.3.19)—(11.3.21) является функция трех переменных $u(x,\,y,\,t)$. При фиксированном $t=t_*$ обычно строят либо трехмерное представление $u(x,\,y,\,t)$, либо на плоскость выводят линии уровня $u(x,\,y,\,t)=\text{const}$ (рис. 11.14). Чтобы построить разностную схему для двумерного уравнения, покроем область сеткой с шагом h по переменным x, y и шагом τ по времени t (рис. 11.15).

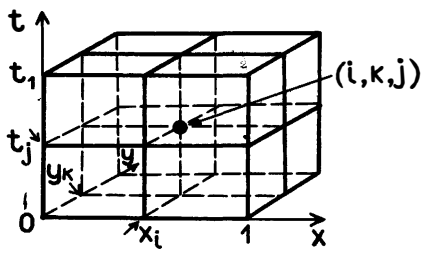

Рис. 11.15

Получим сеточную область

$$D_{h,h,\tau}=\{x_i=ih,\ y_k=kh,\ t_j=j\tau\},$$

где $0\leqslant i\leqslant m,\ 0\leqslant k\leqslant m,\ 0\leqslant j\leqslant p$; $m=1/h,\ p=t_1/\tau$.

Наиболее экономичный по числу арифметических операций метод решения двумерных задач — это метод переменных направлений (или, что то же самое, метод расщепления, метод дробных шагов). Переход с одного временного слоя j на другой $j+1$ разбивают на два этапа:

1) переход на промежуточный временной слой $j+1/2$ с шагом $\tau/2$, решение методом прогонки трехдиагональной системы в направлении оси x;

2) переход на $(j+1)$-й временной слой с промежуточного $(j+1/2)$-го с шагом $\tau/2$, решение методом прогонки трехдиагональной системы в направлении оси y.

Шаблон метода переменных направлений представлен на рис. 11.16. Система разностных уравнений для внутренних узлов сеточной области имеет следующий вид:

$$\frac{u_{i,k,j+1/2}-u_{i,k,j}}{(\tau/2)}=\frac{a^2}{h^2}\left[\left(u_{i+1,k,j+1/2}-2u_{i,k,j+1/2}+u_{i-1,k,j+1/2}\right)+\right.$$

$$+(u_{i,k+1,j}-2u_{i,k,j}+u_{i,k-1,j})]+f_{i,k,j+1/2}, \qquad (11.3.22)$$

$$\frac{u_{i,k,j+1}-u_{i,k,j+1/2}}{(\tau/2)}=\frac{a^2}{h^2}[(u_{i,k+1,j+1}-2u_{i,k,j+1}+u_{i,k-1,j+1})+$$

$$+(u_{i+1,k,j+1/2}-2u_{i,k,j+1/2}+u_{i-1,k,j+1/2})]+f_{i,k,j+1/2}.$$

Здесь введено обозначение $f_{i,k,j+1/2}=f(x_i,\ y_k,\ t_j+\tau/2)$. Кроме уравнений для внутренних узлов задаются значения нулевого временного слоя

$$u_{i,k,0}=\varphi_0(x_i,\ y_k),\ \ 0\leqslant i,\ k\leqslant m, \qquad (11.3.23)$$

и значения на границе сеточного квадрата

$$\begin{cases} u_{0,k,j}=\mu(0,\ y_k,\ t_j),\ u_{m,k,j}=\mu(1,\ y_k,\ t_j), \\ u_{i,0,j}=\mu(x_i,\ 0,\ t_j),\ u_{i,m,j}=\mu(x_i,\ 1,\ t_j). \end{cases} \qquad (11.3.24)$$

В (11.3.24) на дробном временном слое t_j следует заменить на $t_{j+1/2}=t_j+\tau/2$.

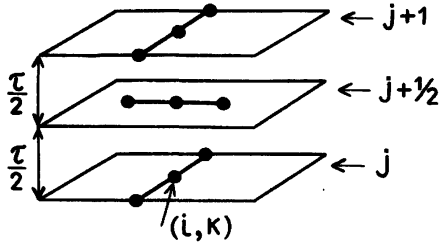

Рис. 11.16

Теорема 11.9. *Пусть решение задачи* (11.3.19)—(11.3.21) $u(x,\ y,\ t)\in C^4[D]$. *Тогда решения разностной схемы метода переменных направлений* $u_{i,k,j}$ *при* $h,\ \tau\to 0$ *сходятся к точному решению в среднеквадратичной норме на временном слое и имеет место оценка*

$$\max_{0\leqslant j\leqslant p}\left(h^2\sum_{i,k=0}^{m-1}(u_{i,k,j}-u(x_i,\ y_k,\ t_j))^2\right)^{1/2}=O(h^2+\tau^2).$$

Оценим объем вычислительных затрат и необходимой памяти ЭВМ для реализации метода переменных направлений. Для того, чтобы перейти с нулевого временного слоя на следующий (1/2)-слой, необходимо решить для $1\leqslant k\leqslant m-1$ трехдиагональную систему:

$$\frac{u_{i,k,1/2}-u_{i,k,0}}{(\tau/2)}=\frac{a^2}{h^2}[(u_{i+1,k,1/2}-2u_{i,k,1/2}+u_{i-1,k,1/2})+$$

$$+(u_{i,k+1,0}-2u_{i,k,0}+u_{i,k-1,0})]+f_{i,k,1/2},$$

или в традиционной записи

$$\frac{\tau a^2}{h^2}u_{i-1,k,1/2}-\left(\frac{2a^2\tau}{h^2}+2\right)u_{i,k,1/2}+\frac{\tau a^2}{h^2}u_{i+1,k,1/2}=F_{i,k,1/2},$$

где

$$F_{i,k,1/2}=-\tau f_{i,k,1/2}-\frac{\tau a^2}{h^2}(u_{i,k+1,0}-2u_{i,k,0}+u_{i,k-1,0})-2u_{i,k,0}$$

— известные величины, вычисляемые по нулевому слою. Чтобы для каждого k, $1\leqslant k\leqslant m-1$, выполнить прогонку, требуется $O(m)$ арифметических операций; таким образом, общее число операций

для определения по $u_{i,k,0}$ значений $u_{i,k,1/2}$ составляет $mO(m)=O(m^2)$. Для вычисления p временных слоев потребуется $\sim pO(m^2)$ арифметических операций. Например, при $m=10^2$, $p=10^3$ число операций $\sim 10^7$ и на ЭВМ с быстродействием 10^6 оп/с требуется ~ 10 с на все вычисления.

Однако, для того чтобы оценить погрешность вычислений (применить правило Рунге), потребуется повторить вычисления с шагом $h/2$, $\tau/2$, а это увеличит число операций в 8 раз. Это составит ~ 80 с и общее время вычислений одного варианта с контролем точности ~ 90 с.

Теперь оценим необходимые ресурсы памяти ЭВМ. Очевидно, что если хранить значения $u_{i,k,j}$ во всех узлах сеточной области $D_{h,h,\tau}$, потребуется $(p+1)(m+1)^2$ ячеек памяти. Это составит $\sim 10^7$ слов, что для небольших ЭВМ превышает объем оперативной памяти. К тому же все временные слои в расчетах хранить не требуется. Обычно в памяти хранят только предыдущий и следующий временные слои, т. е. $u_{i,k,j}$, $u_{i,k,j+1}$, индекс j опускают, старый временной слой обозначают $u_{i,k}$, новый $v_{i,k}$. Тогда объем необходимой памяти для хранения значений $u_{i,k}$, $v_{i,k}$ равен $2(m+1)^2$. Это составит $\sim 2\cdot 10^4$ ячеек, что приемлемо и для малых ЭВМ. Для анализа результатов некоторые временные слои (заданные t_j) выводят на терминал, графопостроители и т. п.

● 11.4. Метод прямых

11.4.1. Введение. Рассматриваемый в этом пункте метод прямых относится к полудискретным методам. В основе полудискретных методов лежит следующая идея: провести дискретизацию дифференциального уравнения с частными производными не полностью по всем независимым переменным, а лишь частично. Например, можно провести дискретизацию только по пространственным переменным (в нестационарных уравнениях), время t оставляется при этом как непрерывная переменная.

Для иллюстрации такого подхода рассмотрим смешанную задачу для уравнения теплопроводности

$$\begin{cases} \dfrac{\partial u}{\partial t}=a^2\dfrac{\partial^2 u}{\partial x^2}, \quad 0\leqslant x\leqslant 1, \quad 0\leqslant t\leqslant 1, \\ u(x,0)=\varphi_0(x), \quad u(0,t)=u(1,t)=0. \end{cases} \tag{11.4.1}$$

Пусть $\{l_0(x), l_1(x), \ldots, l_n(x), \ldots\}$ — базис пространства функций $C^2[0,1]$, обращающихся на концах интервала $0\leqslant x\leqslant 1$ в нуль. Будем искать приближенное решение (11.4.1) в виде

$$u_n(x,t)=\sum_{i=0}^{n}\alpha_i(t)l_1(x). \tag{11.4.2}$$

Чтобы записать уравнения для $\alpha_i(t)$, разобъем интервал $0\leqslant x\leqslant 1$ узлами x_i с постоянным шагом h (рис. 11.17), $h=1/n$. Потребуем,

чтобы на прямых $x=x_i$ функция (11.4.2) удовлетворяла точно дифференциальному уравнению, т. е.

$$\sum_{i=0}^{n} \frac{d\alpha_i}{dt} l_i(x_j) = a^2 \sum_{i=0}^{n} \alpha_i(t) \frac{d^2 l_i}{dx^2}(x_j), \quad 0 \leqslant j \leqslant n.$$

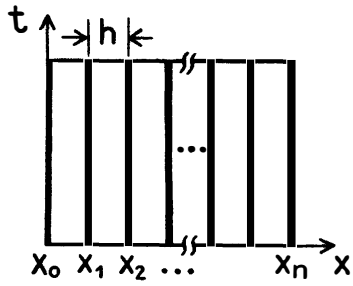

Рис. 11.17

Предположим, что матрица

$$L = l_i(x_j), \quad 0 \leqslant i,j \leqslant n,$$

невырождена; тогда для функций $\alpha_i(t)$ получаем систему линейных обыкновенных дифференциальных уравнений, которую запишем в матричной форме:

$$\frac{d\alpha}{dt} = A\alpha, \tag{11.4.3}$$

где вектор $\alpha=(\alpha_0, \alpha_1, ..., \alpha_n)$, а матрица A определяется формулами

$$A = L^{-1}B, \quad B = \frac{d^2 l_i}{dx^2}(x_j), \quad 0 \leqslant i,j \leqslant n.$$

Потребуем, чтобы функция $u_n(x, t)$ при $t=0$ в узлах x_j совпадала с $\varphi_0(x)$, т. е. удовлетворяла начальному условию. Тогда получим

$$\sum_{i=0}^{n} \alpha_i(0)\, l_i(x_j) = \varphi_0(x_j),$$

или с учетом существования обратной матрицы L^{-1}

$$\alpha(0) = L^{-1}b, \tag{11.4.4}$$

где вектор $b=(\varphi_0(x_0), \varphi_0(x_1), ..., \varphi_0(x_n))$.

Итак, исходная задача сводится к решению задачи Коши для системы линейных уравнений (11.4.3), (11.4.4) с непрерывным временем t, которая получена дискретизацией только по пространственной переменной x.

Если бы система уравнений для $\alpha_i(t)$ была решена аналитически, то функция $u_n(x, t)$, определенная формулой (11.4.2), давала бы n-е приближение к решению исходной задачи. Практически система дифференциальных уравнений по времени t (из-за большой размерности n, а для нелинейных задач из-за нелинейности) должна решаться численно. Поэтому все-таки придется вводить дискретизацию по t; следовательно, термин «полудискретные методы» не совсем точен. Сама идея сведения уравнений с частными производными к системе обыкновенных дифференциальных уравнений весьма полезна. Это объясняется тем, что к полученной системе можно применять разнообразные эффективные процедуры решения обыкновенных дифференциальных уравнений. Например, можно использовать методы с автоматическим выбором шага по точности, которые для обыкновенных дифференциальных уравнений разработаны гораздо глубже, а соответствующие программы входят

в стандартное математическое обеспечение ЭВМ. Последнее замечание может ввести в заблуждение (метод прямых лучше метода конечных разностей) при выборе численного алгоритма решения задач. Следует всегда помнить следующую мысль: нет ни одного метода решения, который был бы хорош для любой задачи; задача выбирает метод, а не метод — задачу.

11.4.2. Метод прямых с конечно-разностной аппроксимацией. Для дискретизации по пространственной переменной можно использовать конечно-разностные аппроксимации. Рассмотрим задачу (11.4.1). Разобьем интервал $0 \leqslant x \leqslant 1$ узлами x_i с шагом h (рис. 11.17). Вдоль прямых $x = x_i$ точное решение задачи — это функция только от t:

$$u(x_i, t) = v_i(t), \quad \frac{\partial u}{\partial t}(x_i, t) = \frac{dv_i(t)}{dt}, \quad 0 \leqslant i \leqslant n.$$

По формуле численного дифференцирования для внутренних линий, $1 \leqslant i \leqslant n-1$, получаем

$$\frac{\partial^2 u}{\partial x^2}(x_i, t) = \frac{v_{i+1}(t) - 2v_i(t) + v_{i-1}(t)}{h^2} + O(h^2).$$

Подставляя эти выражения в исходную задачу и отбрасывая погрешность аппроксимации, получаем систему уравнений метода прямых

$$\left(\frac{h^2}{a^2}\right)\frac{dv_i}{dt} = v_{i+1} - 2v_i + v_{i-1}, \quad 1 \leqslant i \leqslant n-1, \tag{11.4.5}$$

$$v_0(t) = 0, \quad v_n(t) = 0$$

с начальным условием

$$v_i(0) = \varphi_0(x_i), \quad 1 \leqslant i \leqslant n-1. \tag{11.4.6}$$

Система линейных дифференциальных уравнений (11.4.5) имеет трехдиагональную матрицу A правых частей

$$A = \begin{pmatrix} -2 & 1 & & 0 \\ 1 & -2 & 1 & \\ & \ddots & \ddots & \ddots \\ 0 & & 1 & -2 \end{pmatrix};$$

вектор $v = (v_1, \ldots, v_{n-1})$. Найдем собственные значения λ_i матрицы A; $\lambda_i = -\left(2\cos\frac{\pi i}{n} + 2\right)$, $1 \leqslant i \leqslant n-1$. Отсюда следует, что $-4 < \lambda_i < 0$. Аналитическое решение задачи (11.4.5), (11.4.6) имеет вид

$$v(t) = T^{-1}\begin{pmatrix} e^{\frac{\lambda_1 a^2 t}{h^2}} & & 0 \\ & \ddots & \\ 0 & & e^{\frac{\lambda_{n-1} a^2 t}{h^2}} \end{pmatrix} Tv(0), \tag{11.4.7}$$

где матрица T имеет своими столбцами собственные векторы матрицы A, соответствующие собственным значениям λ_i. Элементы матрицы T равны

$$T_{k,i} = \sin(\pi k i / n), \quad 1 \leqslant i, k \leqslant n-1.$$

Прямой подстановкой можно убедиться в справедливости формул для λ_i и $T_{k,i}$.

Самым важным результатом представления решения в форме (11.4.7) является возможность провести анализ коэффициента жесткости S системы уравнений (11.4.5). Как следует из **10.3**, коэффициент жесткости определяется в нашем случае формулой

$$S=S(n)=\frac{\max|\lambda_i|}{\min|\lambda_i|}=\frac{\lambda_1}{\lambda_{n-1}}=\frac{\cos(\pi/n)+1}{\cos\dfrac{\pi(n-1)}{n}+1}=\frac{1+\cos(\pi/n)}{1-\cos(\pi/n)}.$$

При больших значениях n

$$S(n)\simeq\frac{2}{(\pi/n)^2(1/2)}=\frac{4}{\pi^2}n^2.$$

Таким образом, чем мельче шаг h (больше n), тем выше точность приближения метода прямых, но в то же время тем больше коэффициент жесткости системы обыкновенных дифференциальных уравнений метода прямых.

Рассмотренная модельная задача теплопроводности, имеющая точное аналитическое решение метода прямых с конечно-разностной аппроксимацией, содержит основные черты этого метода, которые проявляются в более сложных нелинейных задачах. Важным свойством в этом подходе является высокая жесткость системы обыкновенных дифференциальных уравнений, а следовательно, необходимость решения этих систем при больших n специальными методами (см. 10.3). Более подробно с вопросами сходимости метода прямых и приложениями к другим классам уравнений можно познакомиться в [4, 20].

11.4.3. Применение программы B8A1. Для иллюстрации применения программы B8A1 рассмотрим задачу для одномерного нелинейного уравнения теплопроводности

$$\frac{\partial u}{\partial t}=\frac{\partial}{\partial x}\left(1+u^2+\sin^2 t+x^2\right)\frac{\partial u}{\partial x}+\cos\left(\frac{\partial u}{\partial x}\right)$$

в области $D=\{0\leqslant x\leqslant 1,\quad 0\leqslant t\leqslant 1\}$ с начальным условием

$$u(x,\,0)=0$$

и краевыми условиями

$$\sin t\,u(0,\,t)+\cos t\,\frac{\partial u}{\partial x}(0,\,t)=\mathrm{e}^{-u^2},$$

$$(1+t)\,u(1,\,t)+\mathrm{e}^{-t}\,\frac{\partial u}{\partial x}(1,\,t)=\sin(t+u).$$

Точность интегрирования на одном шаге по времени 10^{-4} (см. описание B8A1). Шаг выдачи на терминал по времени зададим $\Delta t=0{,}1$ по x_i — через десять точек. Число прямых $x_i=\mathrm{const}$ зададим равным 100. Полный контроль точности требует повторного

вызова B8A1 с удвоенным значением *N* (этот фрагмент алгоритма здесь опускается). Программа может иметь следующий вид:

```
      REAL A,B,T0,T1,U(101),E,W(1657),DT
      INTEGER M,N,K,L,I
      EXTERNAL PDEF,BNDY
      DATA A,B,T0,T1/0.,1.,0.,1./,W/101*0./,E/1.E—4/
      DATA M,N,K,L,I/0,101,1657,0,0/,DT/0.1/
C     ВЫЧИСЛЕНИЯ ВРЕМЕННЫХ СЛОЕВ С ПОМОЩЬЮ
C     ПРОГРАММЫ B8A1
      DO 1 J=1,10
      T1=T0+DT
      CALL B8A1(M,A,B,T0,T1,U,N,E,W,K,L,I)
      I=0
C     ВЫВОД НА ТЕРМИНАЛ ВРЕМЕННОГО СЛОЯ
      WRITE (5,2) T1,(U(I),I=1,100,10)
2     FORMAT (2X,'T=',F4.1,10E13.6)
1     CONTINUE
      END
C     ВНЕШНЯЯ ПОДПРОГРАММА, ВЫЧИСЛЯЮЩАЯ
C     ПРАВУЮ ЧАСТЬ ДИФФ. УРАВНЕНИЯ
      SUBROUTINE PDEF(X,T,U,DU,F,Q)
      REAL X,T,U,DU,F,Q
      F=1.+U*U+SIN(T)*SIN(T)+X*X
      Q=COS(DU)
      RETURN
      END
C     ВНЕШНЯЯ ПОДПРОГРАММА, ВЫЧИСЛЯЮЩАЯ
C     ФУНКЦИИ В КРАЕВЫХ УСЛОВИЯХ
      SUBROUTINE BNDY(T,U,I,S,G,R)
      REAL T,U,S,G,R
      INTEGER I
      IF (I.EQ.1) GO TO 1
      S=SIN(T)
      G=COS(T)
      R=EXP(-U*U)
      GO TO 2
1     S=1.+T
      G=EXP(-T)
      R=SIN(T+U)
2     CONTINUE
      RETURN
      END
```

Глава 12

БИБЛИОТЕКА ФОРТРАН-ПРОГРАММ

В настоящей главе описывается библиотека, составленная из программ, входящих обычно в стандартные библиотеки для научно-технических расчетов. Отличием представленных программ является их полная независимость друг от друга. Описание выполнено для простейших вариантов применения программ. Вся библиотека имеет объем ~512 К байт.

● 12.1. Каталог библиотеки

В каталоге приводится список разделов библиотеки. Слева от названия раздела дается обозначение его индекса.

A0. Вычисление элементарных функций
A1. Вычисление специальных функций
A2. Вычисление элементарных функций комплексного переменного
A3. Аппроксимация функций
A4. Интегрирование
A5. Суммирование рядов
A6. Фурье-анализ
A7. Численное дифференцирование
A8. Операции с матрицами и векторами
A9. Решение систем линейных алгебраических уравнений
B0. Вычисление собственных значений и векторов
B1. Линейная оптимизация
B2. Решение линейных интегральных уравнений
B3. Корни полиномов
B4. Корни нелинейных уравнений
B5. Нелинейная оптимизация
B6. Решение обыкновенных дифференциальных уравнений; задача Коши
B7. Решение обыкновенных дифференциальных уравнений; краевая задача
B8. Решение уравнений с частными производными

● 12.2. Содержание разделов

Библиотека состоит из 19 разделов. Каждый раздел содержит подпрограммы, которым присвоено имя, состоящее из индекса раздела и индекса программы в нем, например имя программы A1A2 означает: раздел A1, программа A2. Исключение составляют библиотечные функции фортрана, которые приводятся и с их обычными именами. Всюду, кроме некоторых программ раздела A0, вещественные значения берутся с одинарной точностью.

A0. Вычисление элементарных функций

A0A0:ABS(X)	вещественное абсолютное значение $\|x\|$
A0A1:IABS(I)	целое абсолютное значение $\|i\|$
A0A2:DABS(X)	абсолютное значение с двойной точностью
A0A3:FLOAT(I)	преобразование целой величины в вещественную
A0A4:IFLX(X)	преобразование вещественной величины в целую
A0A5:SNGL(X)	преобразование величины с двойной точностью в вещественную величину с одинарной точностью
A0A6:DBLE(X)	преобразование вещественной величины с одинарной точностью в величину с двойной точностью
A0A7:AINT(X)	выделение целой части вещественной величины с представлением результата в вещественном виде
A0A8:INT(X)	выделение целой части вещественной величины
A0A9:IDINT(X)	выделение целой части величины с двойной точностью
A0B0:AMOD(X,Y)	вычисление вещественного остатка от деления вещественных чисел x/y
A0B1:MOD(X,Y)	вычисление целого остатка от деления целых чисел x/y
A0B2:DMOD(X,Y)	вычисление остатка с двойной точностью от деления чисел с двойной точностью x/y
A0B3:AMAX0(I,J,...)	максимальное (минимальное) вещественное
A0B4:AMIN0(I,J,...)	значение из списка целых чисел
A0B5:AMAX(X,Y,...)	максимальное (минимальное) вещественное
A0B6:AMIN(X,Y,...)	значение из списка вещественных чисел
A0B7:MAX0(I,J,...)	максимальное (минимальное) целое значение
A0B8:MIN0(I,J,...)	из списка целых чисел
A0B9:MAX(X,Y,...)	максимальное (минимальное) целое значение
A0C0:MIN(X,Y,...)	из списка вещественных чисел
A0C1:DMAX(X,Y,...)	максимальное (минимальное) значение с
A0C2:DMIN(X,Y,...)	двойной точностью из списка чисел с двойной точностью

A0C3:SIGN(X,Y,)	вещественное присваивание знака $(\lvert x\rvert \operatorname{sign} y)$
A0C4:ISIGN(I,J)	целое присваивание знака $(\lvert i\rvert \operatorname{sign} j)$
A0C5:DSING(X,Y)	присваивание знака с двойной точностью $(\lvert x\rvert \operatorname{sign} y)$
A0C6:DIM(X,Y)	вещественная положительная разность $(x-\min(x, y))$
A0C7:IDIM(I,J)	целая положительная разность $(i-\min(i, j))$
A0C8:EXP(X)	вещественное значение e^x от вещественного аргумента
A0C9:DEXP(X)	значение e^x с двойной точностью от аргумента с двойной точностью
A0D0:ALOG(X)	вещественное значение $\ln x$ от вещественного аргумента
A0D1:ALOG10(X)	вещественное значение $\lg x$ от вещественного аргумента
A0D2:DLOG(X)	значение $\ln x$ с двойной точностью от аргумента с двойной точностью
A0D3:DLOG10(X)	значение $\lg x$ с двойной точностью от аргумента с двойной точностью
A0D4:SQRT(X)	вещественное значение $\sqrt{x}$ от вещественного аргумента
A0D5:DSQRT(X)	значение $\sqrt{x}$ с двойной точностью от аргумента с двойной точностью
A0D6:SIN(X)	вещественное значение $\sin x$ от вещественного аргумента
A0D7:DSIN(X)	значение $\sin x$ с двойной точностью от аргумента с двойной точностью
A0D8:COS(X)	вещественное значение $\cos x$ от вещественного аргумента
A0D9:DCOS(X)	значение $\cos x$ с двойной точностью от аргумента с двойной точностью
A0E0:TANH(X)	вещественное значение $\operatorname{th} x$ от вещественного аргумента
A0E1:ATAN(X)	вещественное значение $\operatorname{arctg} x$ от вещественного аргумента
A0E2:DATAN(X)	значение $\operatorname{arctg} x$ с двойной точностью от аргумента с двойной точностью

A1. Вычисление специальных функций

A1A0:	интегральная показательная функция $Ei(x)$
A1A1:	гамма функция $\Gamma(x)$
A1A2:	функция ошибок $\operatorname{erf}(x)$
A1A3:	функция Бесселя $Y_0(x)$
A1A4:	функция Бесселя $Y_1(x)$
A1A5:	функция Бесселя $J_0(x)$
A1A6:	функция Бесселя $J_1(x)$

A1A7:	функция Бесселя $K_0(x)$
A1A8:	функция Бесселя $K_1(x)$
A1A9:	функция Бесселя $I_0(x)$
A1B0:	функция Бесселя $I_1(x)$

A2. Вычисление элементарных функций комплексного переменного

A2A0:CABS(Z)	абсолютное значение комплексной $\sqrt{x^2+y^2}$ величины $z=(x, y)$
A2A1:REAL(Z)	выделение действительной части y комплексной величины z
A2A2:AIMAG(Z)	выделение мнимой части у комплексной величины z
A2A3:CMPLX(X,Y)	объединение двух вещественных величин в комплексную $x+iy$
A2A4:CLOG(Z)	логарифм комплексного аргумента $\ln z$
A2A5:CSQRT(Z)	квадратный корень из комплексного аргумента $\sqrt{z}$
A2A6:CSIN(Z)	комплексный синус $\sin z$
A2A7:CCOS(Z)	комплексный косинус $\cos z$
A2A8:CONJG(Z)	комплексное сопряжение $x-iy$

A3. Аппроксимация функций

A3A0:	одномерная интерполяция по схеме Эйткена
A3A1:	двумерная интерполяция бикубическими сплайнами
A3A2:	аппроксимация функции одной переменной полиномами методом наименьших квадратов

A4. Интегрирование

A4A0:	интегрирование одномерных функций на конечном интервале, метод трапеций с уточнением
A4A1:	интегрирование одномерных функций на конечном интервале, метод Гаусса
A4A2:	интегрирование двумерных функций

A5. Суммирование рядов

A5A0:	суммирование числового ряда
A5A1:	вычисление предела последовательности
A5A2:	суммирование рядов Чебышева

A6. Фурье-анализ

A6A0:	Фурье-преобразование N вещественных чисел
A6A1:	Фурье-преобразование N комплексных чисел

A7. Численное дифференцирование

A7A0: дифференцирование таблично заданной функции с переменным шагом

A7A1: дифференцирование таблично заданной функции с постоянным шагом

A7A2: диффренцирование функции, заданной подпрограммой

A8. Операции с матрицами и векторами

A8A0: сложение двух матриц

A8A1: матричное умножение

A8A2: транспонирование матрицы

A8A3: скалярное произведение векторов

A8A4: вычисление определителя вещественной симметричной положительно определенной матрицы

A8A5: вычисление определителя вещественной матрицы

A9. Решение систем линейных алгебраических уравнений

A9A0: решение вещественной системы линейных уравнений

A9A1: решение вещественной системы линейных уравнений с симметричной положительно определенной матрицей

B0. Вычисление собственных значений и векторов

B0A0: вычисление собственных значений вещественной симметричной матрицы

B0A1: вычисление собственных значений и собственных векторов вещественной симметричной матрицы

B0A2: вычисление собственных значений вещественной матрицы

B0A3: вычисление собственных значений и собственных векторов вещественной матрицы

B1. Линейная оптимизация

B1A0: решение задачи линейной оптимизации симплекс-методом

B2. Решение линейных интегральных уравнений

B2A0: решение линейного интегрального уравнения Фредгольма второго рода

B3. Корни полиномов

B3A0: определение корней полинома с вещественными коэффициентами

B4. Корни нелинейных уравнений

B4A0: поиск корня непрерывной функции одной переменной без вычисления производной

B4A1: поиск корня функции одной переменной методом Ньютона

B4A2: решение системы нелинейных уравнений

B5. Нелинейная оптимизация

B5A0: минимизация функции одной переменной

B5A1: минимизация функции нескольких переменных

B6. Решение обыкновенных дифференциальных уравнений; задача Коши

B6A0: интегрирование системы обыкновенных дифференциальных уравнений методом Рунге—Кутта

B6A1: интегрирование системы жестких обыкновенных дифференциальных уравнений

B7. Решение обыкновенных дифференциальных уравнений; краевая задача

B7A0: решение краевой задачи для системы линейных обыкновенных дифференциальных уравнений

B7A1: решение краевой задачи для нелинейных обыкновенных дифференциальных уравнений

B8. Решение уравнений с частными производными

B8A0: решение эллиптического уравнения в прямоугольнике разностным методом

B8A1: решение одномерного параболического уравнения методом прямых

● 12.3. Описание программ

12.3.1. Вычисление элементарных функций. Программы A0A0—A0E2 являются библиотечными функциями фортрана, обращение к которым осуществляется с учетом значений аргумента (целое, вещественное, с двойной точностью) и функции. Например:

```
1) REAL X,Y
   X=0.4
   Y=EXP(X)
```

```
2) DOUBLE PRECISION X,Y
   X=1.0+D-03
   Y=DEXP(X)
```

```
3) REAL X
   INTEGER K
   X=3. 4
   K=INT(X)
```

12.3.2. Вычисление специальных функций. Определение специальных функций и формулы для их вычисления можно найти в [22,23].

A1A0: *интегральная показательная функция* $\mathrm{E}i(x)$

Программа вычисляет

$$Ei(x)=\int_x^{\infty}(\mathrm{e}^{-s}/s)\,ds,\quad x>0,$$

FUNCTION A1A0(X,I)

Параметры входные:

X — вещественный аргумент функции,
I — целое число — индекс ошибки, до обращения следует положить I=0;

выходные:

A1A0 — *значение* $Ei(x)$;

$$\mathrm{I}=\begin{cases}0 — \text{нет ошибок,}\\ 1 — \text{аргумент } X\leqslant 0, \text{ для которого функция } Ei(x) \text{ не определена.}\end{cases}$$

A1A1: *гамма-функция* $\Gamma(x)$

Программа вычисляет

$$\Gamma(x)=\begin{cases}\displaystyle\int_0^{\infty}(\mathrm{e}^{-s}/s^{x-1})\,ds,\quad x>0,\\ \displaystyle\lim_{n\to\infty}\frac{n!\,n^{x-1}}{x(x+1)\ldots(x+n-1)},\quad \text{любые } x.\end{cases}$$

FUNCTION A1A1(X,I)

Параметры входные:

X — вещественный аргумент функции,
I — индекс ошибки, до обращения положить I=0;

выходные:

A1A1: — *значение* $\Gamma(x)$;

$$I=\begin{cases}0 — \text{нет ошибок},\\ 1 — \text{аргумент слишком велик},\\ 2 — \text{аргумент слишком большое отрицательное число},\\ 3 — \text{аргумент близок к нулю},\\ 4 — \text{аргумент — целое отрицательное число}.\end{cases}$$

A1A2: *функция ошибок* $\operatorname{erf}(x)$

Программа вычисляет

$$\operatorname{erf}(x)=\frac{2}{\sqrt{\pi}}\int_0^x \exp(-s^2)\,ds,$$

FUNCTION A1A2(X,I)

Параметры входные:

X — вещественный аргумент функции,
I — индекс ошибки, до обращения положить I=0;

выходные:

A1A2 — значение $\operatorname{erf}(x)$;
I=0 — нет ошибок.

A1A3: *функция Бесселя* $Y_0(x)$

Программа вычисляет $Y_0(x)$ для $x>0$

FUNCTION A1A3(X,I)

Параметры входные:

X — вещественный аргумент функции,
I — индекс ошибки, до обращения положить I=0;

выходные:

A1A3: — *значение* $Y_0(x)$;

$$I=\begin{cases}0 — \text{нет ошибок},\\ 1 — \text{аргумент слишком велик},\\ 2 — \text{аргумент отрицателен, } Y_0(x) \text{ не определена}.\end{cases}$$

A1A4: *функция Бесселя* $Y_1(x)$

Программа вычисляет $Y_1(x)$ для $x>0$.

FUNCTION A1A4(X,I)

Параметры входные:

X — вещественный аргумент функции,
I — индекс ошибки, до обращения положить I=0;

выходные:

A1A4: — *значение* $y_1(x)$;

$$I=\begin{cases}0 — \text{нет ошибок},\\ 1 — \text{аргумент слишком велик},\\ 2 — \text{аргумент отрицателен, } y_1(x) \text{ не определена},\\ 3 — \text{аргумент близок к нулю}.\end{cases}$$

A1A5: *функция Бесселя* $J_0(x)$

Программа вычисляет $J_0(x)$ для $x \geqslant 0$, в интервале $x<0$ следует использовать тождество $J_0(-x)=J_0(x)$.

FUNCTION A1A5(X,I)

Параметры входные:

X — вещественный аргумент функции,
I — индекс ошибки, до обращения положить I=0;

выходные:

A1A5: — *значение* $J_0(x)$;

$$I=\begin{cases}0 — \text{нет ошибок},\\ 1 — \text{аргумент слишком велик}.\end{cases}$$

A1A6: *функция Бесселя* $J_1(x)$

Программа вычисляет $J_1(x)$ для $x \geqslant 0$, в интервале $x<0$ следует использовать тождество $J_1(-x)=-J_1(x)$.

FUNCTION A1A6(X,I)

Параметры входные:

X — вещественный аргумент функции,
I — индекс ошибки, до обращения положить I=0;

выходные:

A1A6: — *значение* $J_1(x)$;

$$I=\begin{cases}0 — \text{нет ошибок},\\ 1 — \text{аргумент слишком велик}.\end{cases}$$

A1A7: *функция Бесселя* $K_0(x)$

Программа вычисляет $K_0(x)$ для $x>0$.

FUNCTION A1A7(X,I)

Параметры входные:

X — вещественный аргумент функции,
I — индекс ошибки, до обращения положить I=0;

выходные:

A1A7: — *значение* $K_0(x)$;

$$I=\begin{cases}0 — \text{нет ошибок},\\ 1 — \text{аргумент меньше нуля}, K_0(x) — \text{не определена}.\end{cases}$$

A1A8: *функция Бесселя* $K_1(x)$

Программа вычисляет $K_1(x)$ для $x>0$.

FUNCTION A1A8(X,I)

Параметры входные:

X — вещественный аргумент функции,
I — индекс ошибки, до обращения положить I=0;

выходные:

A1A8: — *значение* $K_1(x)$;

$$I=\begin{cases}0 — \text{нет ошибок},\\ 1 — \text{аргумент меньше нуля}, K_1(x) — \text{не определена},\\ 2 — \text{аргумент близок к нулю}.\end{cases}$$

A1A9: *функция Бесселя* $I_0(x)$.

Программа вычисляет $I_0(x)$ для $x\geqslant 0$, в интервале $x<0$ следует использовать тождество $I_0(-x)=I_0(x)$.

FUNCTION A1A9(X,I)

Параметры входные:

X — вещественный аргумент функции,
I — индекс ошибки, до обращения положить I=0;

выходные:

A1A9: — *значение* $I_0(x)$

$$I=\begin{cases}0 — \text{нет ошибок},\\ 1 — \text{аргумент слишком велик}.\end{cases}$$

A1B0: *функция Бесселя* $I_1(x)$.

Программа вычисляет $I_1(x)$ для $x \geqslant 0$, в интервале $x<0$ следует использовать тождество $I_1(-x)=-I_1(x)$.

FUNCTION A1B0(X,I)

Параметры входные:

X — вещественный аргумент функции,
I — индекс ошибки, до обращения положить I=0;

выходные:

A1B0: — *значение* $I_1(x)$

$$\mathrm{I}=\begin{cases}0 — \text{нет ошибок,}\\ 1 — \text{аргумент слишком велик.}\end{cases}$$

12.3.3. Вычисление элементарных функций комплексного переменного. Программы A2A0 — A2A8 являются библиотечными функциями фортрана, обращение к которым осуществляется с соответствующим описанием типа аргумента и функции, например:

```
COMPLEX Z,W          REAL W
Z=(1.7,-3.2)         COMPLEX Z
W=CCOS(Z)            Z=(1.7,-3.2)
                     W=AIMAG(Z)
```

12.3.4. Аппроксимация функций

A3A0: *одномерная интерполяция по схеме Эйткена*

Программа вычисляет по таблично заданной функции $y_i=f(x_i)$ в узлах $x_i, 1 \leqslant i \leqslant N$, значение функции $f(z)$ в точке $z \neq x_i$ по схеме Эйткена с заданной точностью ε.

SUBROUTINE A3A0(N,X,Y,Z,E,F,P,I)

Параметры входные:

N — количество узлов,
X — одномерный массив x_i, $1 \leqslant i \leqslant N$,
Y — одномерный массив y_i, $1 \leqslant i \leqslant N$,
Z — значение z, где функция вычисляется интерполированием,
E — заданная точность ε,
F — одномерный рабочий массив F_i, $1 \leqslant i \leqslant N$;

выходные:

P — значение $f(z)$,
I — индекс ошибки,

$$\mathrm{I}=\begin{cases}0 — \text{нет ошибок,}\\ 1 — \text{количество узлов } N \text{ недостаточно для достижения заданной точности.}\end{cases}$$

A3A1: *двумерная интерполяция бикубическими сплайнами*

Программа интерполирует в точке плоскости $(\bar{x}, \bar{y})$ двумерную функцию $f(x, y)$, заданную таблично на прямоугольной сетке, бикубическими сплайнами. Интерполирование по схеме: при фиксированных y_j, $1 \leqslant j \leqslant M$, для x_i, $1 \leqslant i \leqslant N$, определяются интерполяционные кубические сплайны $S_1(x_i, y_j)$ и интерполяционные значения $S_1(\bar{x}, y_j)$, $1 \leqslant j \leqslant M$; затем находятся интерполяционный сплайн $S_{0,1}(\bar{x}, y_j)$ и интерполированное значение $S_{0,1}(\bar{x}, \bar{y})$. Та же схема повторяется с изменением порядка координат: сначала фиксируются x_i, находится $S_0(x_i, \bar{y})$, затем значение $S_{1,0}(\bar{x}, \bar{y})$.

SUBROUTINE A3A1(XC,YC,X,Y,F,S01,S10,I,W1,W2,W3,W4,J,M,N)

Параметры входные:

XC — вещественное, значение $\bar{x}$,
YC — вещественное, значение $\bar{y}$,
X — вещественный одномерный массив, размерность N, содержит $x_1 < x_2 < \ldots < x_N$,
Y — вещественный одномерный массив, размерность M, содержит $y_1 < y_2 < \ldots < y_M$
F — вещественный двумерный массив, размерность (N, M), содержит значения $f(x_i, y_j)$,
I — целое, индекс ошибки перед обращением положить I=0, W1, W2, W3, W4 — вещественные одномерные массивы, размерность J, рабочие массивы,
J — целое, $\max(N, M)$
M — целое, число узлов по оси y,
N — целое, число узлов по оси x;

выходные:

S01 — вещественное, интерполяционное значение $S_{0,1}$,
S10 — вещественное, интерполяционное значение $S_{1,0}$;

$$\mathrm{I} = \begin{cases} 0 — \text{нет ошибок}, \\ 1 — \text{нарушено условие } x_1 \leqslant \bar{x} \leqslant x_N,\ y_1 \leqslant \bar{y} \leqslant y_M. \end{cases}$$

A3A2: *аппроксимация функции одной переменной полиномами методом наименьших квадратов*

Программа определяет методом наименьших квадратов полином вида

$$P_i(x) = 0{,}5a_{i+1,1}\,T_0(x) + \sum_{j=1}^{i} a_{i+1,j+1}\,T_j(x),$$

$(0 \leqslant i \leqslant K,\ -1 \leqslant x \leqslant +1,\ T_j(x)$ — полиномы Чебышева), аппроксимирующий таблично заданную функцию $y(x)$:

аргумент x	$x_1\ x_2\ \ldots\ x_M$
значение $y(x)$	$y_1\ y_2\ \ldots\ y_M$

Если аргумент x лежит в интервале $a \leqslant x \leqslant b$, следует выполнить замену

$$\tilde{x} = (2x - b - a)/(b - a), \quad -1 \leqslant \tilde{x} \leqslant +1.$$

Невязка, подлежащая минимизации, задается формулой

$$\varepsilon_{i+1} = \left(\frac{1}{M - i - 1} \sum_{l=1}^{M} w_l (y_l - P_i(x_l))^2 \right)^{1/2}, \quad 0 \leqslant i \leqslant K,$$

где w_l — весовые коэффициенты, $w_l > 0$; если $M = i + 1$, $\varepsilon_{i+1} = 0$.

На входе программы задаются: максимальная степень полинома K, массивы $(x_i,\ y_i)$, веса w_l; на выходе получаются массивы коэффициентов $a_{i+1,\,j+1}\ (0 \leqslant i \leqslant K,\ 0 \leqslant j \leqslant i)$ и массив погрешностей $\varepsilon_{i+1}\ (0 \leqslant i \leqslant K)$. Вычисление $P_i(x)$ с полученными коэффициентами в любой точке $x, -1 \leqslant x \leqslant +1$, можно выполнить с помощью программы A5A2.

SUBROUTINE A3A2(M,K1,N,X,Y,W,W1,W2,A,E,I)

Параметры входные:

M — целое, число точек в таблице, $M \geqslant 2$,
K1 — целое, равное $K+1$, где K — максимальная степень полинома,
N — целое, число строк массива A, $N \geqslant K1$,
X — вещественный одномерный массив, размерность M, содержит массив $(x_1,\ x_2,\ \ldots,\ x_M)$,
Y — вещественный одномерный массив, размерность M, содержит массив $(y_1,\ y_2,\ \ldots,\ y_M)$,
W — вещественный одномерный массив, размерность M, содержит массив весовых коэффициентов $(w_1,\ w_2,\ \ldots,\ w_M)$,
W1 — вещественный двумерный массив, размерность (3, M), рабочий массив,
W2 — вещественный двумерный массив, размерность (2, $K1$), рабочий массив,
I — целое, индекс ошибки, перед обращением положить I=0;

выходные:

E — вещественный одномерный массив, размерность $K1$, содержит невязки ε_{i+1}; $E(I+1) = \varepsilon_{i+1}\ (0 \leqslant i \leqslant K)$,
A — вещественный двумерный массив, размерность (N, p), $p \geqslant K1$ содержит коэффициенты $a_{i+1,\,j+1}$; $A(I+1,\ J+1) = a_{i+1,\,j+1}$;

$$I = \begin{cases} 0 — \text{нет ошибок}, \\ 1 — \text{параметры лежат вне допустимых интервалов}. \end{cases}$$

12.3.5. Интегрирование

A4A0: *интегрирование одномерных функций на конечном интервале, метод трапеций с уточнением*

Программа вычисляет приближенное значение

$$y=\int_a^b f(x)\,dx$$

последовательно по формуле трапеций с удваиванием числа узлов и уточнением

SUBROUTINE A4A0(A,B,E,N,F,Y,I,W)

Параметры входные:

A — нижний предел интегрирования a,
B — верхний предел интегрирования b,
E — абсолютная погрешность,
N — максимальное число делений пополам отрезка $[a, b]$
F — имя внешней функции-подпрограммы, вычисляющей $f(x)$
W — рабочий одномерный массив, размерность N;

выходные:

Y — значение интеграла y,
I — индекс ошибки;

$$I=\begin{cases}0 — \text{нет ошибок},\\ 1 — \text{точность не достигается},\\ 2 — \text{точность не достигается, требуется увеличить } N.\end{cases}$$

A4A1: *интегрирование одномерных функций на конечном интервале, метод Гаусса*

Программа вычисляет приближенное значение

$$y=\int_a^b f(x)\,dx$$

по формулам Гаусса с выбором числа узлов по заданной абсолютной и относительной точности.

SUBROUTINE A4A1(F,A,B,E,D,Y,E1)

Параметры входные:

F — имя внешней функции-подпрограммы, вычисляющей $f(x)$,
A — нижний предел интегрирования a,
B — верхний предел интегрирования b,
E — заданная абсолютная погрешность,
D — заданная относительная погрешность;

выходные:

Y — значение интеграла y,

E1 — достигнутая абсолютная погрешность $|y-Y|\leqslant E1$. Если $E1>\max(E,D|Y|)$, заданная точность не достигается с максимальным числом узлов.

A4A2: *интегрирование двумерных функций*

Программа вычисляет приближенное значение

$$z=\int_a^b dy \int_{\varphi_1(y)}^{\varphi_2(y)} f(x, y)\,dx.$$

Внутренний интеграл $v(y)=\int_{\varphi_1(y)}^{\varphi_2(y)} f(x, y)\,dx$ и внешний $z=\int_a^b v(y)\,dy$ вычисляются по формулам Гаусса с выбором числа узлов по заданной точности.

SUBROUTINE A4A2(A,B,F1,F2,F,E,Z,N,I)

Параметры входные:

A — нижний предел a интегрирования по y,

B — верхний предел b интегрирования по y,

F1 — имя внешней функции-подпрограммы, вычисляющей $\varphi_1(y)$,

F2 — имя внешней функции-подпрограммы, вычисляющей $\varphi_2(y)$,

F — имя внешней функции-подпрограммы, вычисляющей $f(x, y)$,

E — заданная абсолютная погрешность,

I — индекс ошибки, до обращения положить I=0;

выходные:

Z — значение интеграла,

N — число узлов, в которых вычислялась $f(x, y)$;

I = 0 — нет ошибок,

I = 1 — точность не достигается с максимальным числом узлов.

12.3.6. Суммирование рядов

A5A0: *суммирование числового ряда*

Программа вычисляет сумму ряда

$$S=\sum_{k=1}^{\infty} u_k$$

по заданной относительной точности посредством преобразования Эйлера, ускоряющего сходимость.

SUBROUTINE A5A0(U,S,N,E,I)

Параметры входные:

U — имя внешней функции-подпрограммы, вычисляющей k-й член ряда по заданному k
REAL FUNCTION U(K)
INTEGER K

N — целое, максимальное число слагаемых в частичной сумме ряда, $N \geqslant 1$,

E — вещественное, относительная погрешность вычисления S;

выходные:

S — вещественное приближенное значение суммы ряда,

I — индекс ошибки;

$$I = \begin{cases} 0 — \text{нет ошибок} \\ 1 — \text{точность не достигается с заданным } N. \end{cases}$$

A5A1: *вычисление предела последовательности*

Программа вычисляет предел последовательности

$$S = \lim_{k \to \infty} u_k$$

с помощью ускоряющего сходимость преобразования по заданной относительной точности.

SUBROUTINE A5A1(U,N,S,E,I)

Параметры входные:

U — вещественный одномерный массив, размерность N, содержит $(u_1, u_2, \ldots, u_N)$,

N — целое, размерность массива u, $N \geqslant 10$,

E — вещественное, относительная погрешность вычисления S;

выходные:

S — вещественное, приближенное значение предела,

I — индекс ошибки;

$$I = \begin{cases} 0 — \text{нет ошибок} \\ 1 — \text{точность не достигается с заданным } N. \end{cases}$$

A5A2: *суммирование рядов Чебышева*

Программа вычисляет сумму отрезка ряда Чебышева в заданной точке x, $-1 \leqslant x \leqslant +1$, по заданным коэффициентам a_i, $1 \leqslant i \leqslant N$, следуя формулам

$$S=\begin{cases} 0{,}5a_1+\sum\limits_{i=2}^{N} a_i T_{i-1}(x), & m=1, \\ 0{,}5a_1+\sum\limits_{i=2}^{N} a_i T_{2i-2}(x), & m=2, \\ \sum\limits_{i=1}^{N} a_i T_{2i-1}(x) & m=3, \end{cases}$$

$m=1$ соответствует общему случаю, $m=2$—сумма полиномов четной степени, $m=3$—нечетной.

FUNCTION A5A2(X,A,N,M)

Параметры входные:

X — вещественное, аргумент x,
A — вещественный одномерный массив, размерность N, содержит коэффициенты (a_1, a_2, ..., a_N),
N — целое, размерность массива A,
M — целое, индекс типа ряда m;

выходные:

A5A2 — вещественное, содержит значение S.

12.3.7. Фурье-анализ. В этом разделе представлены программы, вычисляющие дискретное Фурье-преобразование N комплексных чисел

$$z_j=x_j+iy_j,\quad 0\leqslant j\leqslant N-1$$

по формулам

$$w_k=\frac{1}{\sqrt{N}}\sum_{j=0}^{N-1} z_j\exp\left(-i\frac{2\pi jk}{N}\right),\quad 0\leqslant k\leqslant N-1, \qquad (12.3.1)$$

или в вещественной форме

$$w_k=a_k+ib_k,$$

$$a_k=\frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}\left[x_j\cos\left(\frac{2\pi jk}{N}\right)+y_j\sin\left(\frac{2\pi jk}{N}\right)\right],$$

$$b_k=\frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}\left[y_j\cos\left(\frac{2\pi jk}{N}\right)-x_j\sin\left(\frac{2\pi jk}{N}\right)\right].$$

Обратное дискретное Фурье-преобразование задается формулой

$$z_j=\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} w_k\exp\left(i\frac{2\pi jk}{N}\right),\quad 0\leqslant j\leqslant N-1.$$

Отсюда следует, что комплексно-сопряженные величины $\bar{z}_j$ есть прямое дискретное Фурье-преобразование от $\bar{w}_k$. Следовательно, обратное дискретное Фурье-преобразование для w_k получается переходом к комплексному сопряжению $\bar{w}_k$, выполнением прямого

преобразования и, наконец, комплексным сопряжением результата. Поэтому ниже рассматривается только прямое преобразование. Используются алгоритмы быстрого Фурье-преобразования.

A6A0: *Фурье-преобразование N вещественных чисел*

Программа вычисляет Фурье-преобразование N вещественных чисел x_j, $0 \leqslant j \leqslant N-1$ по формуле

$$w_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \exp\left(-i \frac{2\pi jk}{N}\right), \quad 0 \leqslant k \leqslant N-1. \tag{12.3.2}$$

Величины w_k комплексные, обладающие свойством

$$w_{N-k} = \bar{w}_k.$$

SUBROUTINE A6A0(X,N,I)

Параметры входные:

X — вещественный одномерный массив, размерность N, компонента $X(j+1)$ должна содержать x_j,
N — размерность X, целое число, $N=2^p$, $p \leqslant 20$,
I — индекс ошибки, до обращения положить I=0;

выходные:

X — содержит a_k, $b_к$ для w_k, причем $a_k = X(k)$, $0 \leqslant k \leqslant N/2$, $b_k = X(k)$, $N/2 < k \leqslant N-1$; остальные компоненты получаются из равенств $a_{N-k} = a_k$, $b_{N-k} = -b_k$;
I = 0 — нет ошибок,
I = 1 — N слишком велико, $p > 20$.

A6A1: *Фурье-преобразование N комплексных чисел*

Программа вычисляется преобразованием чисел z_j, $0 \leqslant j \leqslant N-1$, по формулам (12.3.1).

SUBROUTINE A6A1(X,Y,N,I)

Параметры входные:

X — вещественный одномерный массив, размерность N, должен содержать $X(j+1) = \mathrm{Re}\, z_j = x_j$, $0 \leqslant j \leqslant N-1$,
Y — вещественный одномерный массив, размерность N, должен содержать $Y(j+1) = \mathrm{Im}\, z_j = y_j$, $0 \leqslant j \leqslant N-1$,
N — размерность, целое число $N=2^p$, $p \leqslant 20$,
I — индекс ошибки, до обращения положить I=0;

выходные:

X — содержит $X(k+1) = \mathrm{Re}\, w_k = a_k$, $0 \leqslant k \leqslant N-1$,
Y — содержит $Y(k+1) = \mathrm{Im}\, w_k = b_k$, $0 \leqslant k \leqslant N-1$;

$$I=\begin{cases}0 — \text{нет ошибок},\\ 1 — N \text{ слишком велико}, p>20.\end{cases}$$

12.3.8. Численное дифференцирование

A7A0: *дифференцирование таблично заданной функции с переменным шагом*

Программа вычисляет численно первую производную $y(x)$, используя интерполирование по трем последовательным точкам таблицы $(x_i,\ y_i)$:

аргумент	$x_1\ x_2 \dots x_N$
значение функции	$y_1\ y_2 \dots y_N$
значение производной	$z_1\ z_2 \dots z_N$

Ошибка аппроксимации ε следующая:

$$\varepsilon=\max_{1\leqslant i\leqslant N}\left|\frac{dy}{dx}(x_i)-z_i\right|\leqslant\frac{1}{6}\max_i(x_{i+1}-x_i)^2\max_{x_1\leqslant x\leqslant x_N}\left|\frac{d^3y}{dx^3}(x)\right|.$$

SUBROUTINE A7A0(X,Y,Z,N,I)

Параметры входные:

X — вещественный одномерный массив, размерность N, содержит массив x_i

Y — вещественный одномерный массив, размерность N, содержит массив y_i

N — целое, размерность массивов X, Y, Z;

выходные:

Z — вещественный одномерный массив, размерность N, содержит массив z_i,

I — индекс ошибки,

$$I=\begin{cases}0 — \text{нет ошибок},\\ -1 — N<3,\\ >0 — \text{совпадают узлы } x_i=x_j,\ i\neq j.\end{cases}$$

A7A1: *дифференцирование таблично заданной функции с постоянным шагом*

Программа вычисляет численно первую производную $y(x)$, используя интерполирование по пяти последовательным точкам таблицы $(x_i,\ y_i)$, $x_{i+1}-x_i=h$; по массиву y_i, шагу h находится численное значение z_i производной в узлах. Ошибка аппроксимации ε следующая:

$$\varepsilon=\max_{1\leqslant i\leqslant N}\left|\frac{dy}{dx}(x_i)-z_i\right|\leqslant\frac{h^4}{5}\max_{x_1\leqslant x\leqslant x_N}\left|\frac{d^5y}{dx^5}(x)\right|.$$

Ошибка округления $\varepsilon_1\leqslant 11\,\gamma/h$, где γ — абсолютная ошибка округления при представлении чисел y_i в памяти ЭВМ.

SUBROUTINE A7A1(H,Y,Z,N,I)

Параметры входные:

H — вещественное, шаг таблицы h,
Y — вещественный одномерный массив, размерность N, содержит y_i,
N — целое, размер таблицы;

выходные:

Z — вещественный одномерный массив, размерность N, содержит массив z_i
I — индекс ошибки;

$$I=\begin{cases} 0 — \text{нет ошибок}, \\ -1 — N<5, \\ 1 — H=0. \end{cases}$$

A7A2: *дифференцирование функции, заданной подпрограммой*

SUBROUTINE A7A2(X,N,H,D,E,F,I)

Параметры входные:

X — вещественное, значение x, в которой вычисляется $\frac{df}{dx}(x)$
N — целое, порядок производной $0<N\leqslant 14$,
H — вещественное, задаваемый шаг для вычисления производных,
F — имя внешней функции, вычисляющей значение $f(x)$
REAL FUNCTION F(X)
REAL X
I — индекс ошибки, перед входом положить I=0;

выходные:

D — вещественный одномерный массив, размерность 14, содержит $D(I)=\frac{d^i f}{dx^i}(x)$, $1\leqslant i\leqslant N$,

E — вещественный одномерный массив, размерность 14, содержит абсолютные погрешности в вычислении i-й производной;

$$I=\begin{cases} 0 — \text{нет ошибок}, \\ 1 — N=0, \text{ или } H=0. \end{cases}$$

12.3.9. Операции с матрицами и векторами

A8A0: *сложение двух матриц*

Программа выполняет сложение матриц B и C по формуле

$$A_{i,j}=B_{i,j}+C_{i,j},\quad 1\leqslant i\leqslant M,\quad 1\leqslant j\leqslant N$$

SUBROUTINE A8A0(A,B,C,M,N,I)

Параметры входные:

B — вещественный двумерный массив, размерность (M, N),
C — вещественный двумерный массив, размерность (M, N),
M — число строк M матриц B и C,
N — число столбцов N матриц B и C,
I — индекс ошибки, до обращения положить I=0;

выходные:

A — вещественный двумерный массив, размерность (M, N), сумма матриц B и C;

$$\mathrm{I}=\begin{cases}0 — \text{нет ошибок},\\ 1 — M \text{ или } N \text{ меньше либо равны } 0.\end{cases}$$

A8A1: *умножение двух матриц*

Программа выполняет умножение матриц B и C и, в зависимости от заданного варианта, результат помещается в A или B или C.

$$A_{i,j}=\sum_{k=1}^{M} B_{i,k}C_{k,j},\quad 1\leqslant i\leqslant N,\quad 1\leqslant j\leqslant P.$$

SUBROUTINE A8A1(A,B,C,N,P,M,Z,K,L,I)

Параметры входные:

B — вещественный двумерный массив, размерность (N, M),
C — вещественный двумерный массив, размерность (M, P),
N — число строк массивов B и A,
P — число столбцов массивов A и C,
M — число строк массива C и столбцов массива B,
Z — вещественный одномерный массив, размерность K,
K — целое, если $L=1$, то $K=1$; в других случаях $K=M$,
L — целое, номер варианта, определяет, куда поместить результат, если:

$L=1$, $BC\to A$,
$L=2$, $BC\to B$,
$L=3$, $BC\to C$;

I — индекс ошибки, до обращения положить I=0;

выходные:

A — результат умножения, если $L=1$,
B — результат умножения, если $L=2$,
C — результат умножения, если $L=3$;

$$\mathrm{I}=\begin{cases} 0 — \text{нет} & \text{ошибок}, \\ 1 — N & \text{или} \quad P \text{ или } M \leqslant 0, \\ 2 — L=2 & \text{и} \quad M \neq P, \\ 3 — L=3 & \text{и} \quad N \neq M, \\ 4 — L=1 & \text{и} \quad K < M. \end{cases}$$

A8A2: *транспонирование матрицы*

Программа производит транспонирование матрицы A размером $M \times N$, записанной в одномерный массив по столбцам. Результат A' содержится в исходном массиве.

SUBROUTINE A8A2(A,M,N,K,L,J,I)

Параметры входные:

A — вещественный одномерный массив, размерность (M, N),
M — целое, равное M — число строк A,
N — целое, равное N — число столбцов A,
K — целое, равное $M \times N$,
L — целый одномерный массив, размерность J, рабочий массив,
J — целое, до обращения присвоить $[(M+N)/2]$,
I — индекс ошибки, до обращения положить I=0;

выходные:

A — вещественный одномерный массив, размерность (M, N), содержит транспонированную матрицу, записанную по столбцам;

$$\mathrm{I}=\begin{cases} 0 — \text{нет ошибок}, \\ 1 — K \neq M \times N \end{cases}$$

A8A3: скалярное произведение векторов

Программа вычисляет $y=\sum_{i=1}^{N} a_i b_i$ — скалярное произведение векторов a и b.

FUNCTION A8A3(A,B,N)

Параметры входные:

A — вещественный одномерный массив, размерность N, содержит вектор a,
B — вещественный одномерный массив, размерность N, содержит вектор b,
N — целое, размерность векторов;

выходные:

A8A3 — значение y.

A8A4: *вычисление определителя вещественной симметричной положительно определенной матрицы*

Программа вычисляет определитель матрицы A разложением ее в произведение $A=LL'$, где L — нижняя треугольная матрица, L' — транспонированная к L. Определитель есть сумма квадратов диагональных элементов L.

SUBROUTINE A8A4(A,K,N,D,W,I)

Параметры входные:

A — вещественный двумерный массив, размерность $K\times P$, где $P\geqslant N$, для вычислений нужна только верхняя треугольная часть, нижняя поддиагональная часть используется как рабочий массив,

K — целое, число строк A, описанное в вызывающей программе, $K\geqslant N$; в простом варианте положить $P=K=N$,

N — целое, порядок матрицы A, для которой вычисляется определитель,

W — вещественный одномерный массив, размерность N, рабочий массив,

I — индекс ошибки, до обращения положить I=0;

выходные:

D — значение $\det A$;

I= 0 — нет ошибок,
1 — матрица A не положительно определена,
2 — определитель слишком велик,
3 — определитель слишком мал.

A8A5: *вычисление определителя вещественной матрицы*

Программа вычисляет определитель матрицы A с помощью разложения на множители $A=LU$.

SUBROUTINE A8A5(A,K,N,D,W,I)

Параметры входные:

A — вещественный двумерный массив, размерность (K, P), где $P\geqslant N$, содержит матрицу A,

K — целое число строк A, описанное в вызывающей программе, $K\leqslant N$; в простом варианте положить $P=K=N$,

N — целое, порядок матрицы A, для которой вычисляется определитель,

W — вещественный одномерный массив, размерность N, рабочий массив,

I — индекс ошибки, до обращения положить I=0;

выходные:

D — значение $\det A$;

$$I=\begin{cases}0 — \text{нет ошибок},\\ 1 — \text{матрица } A \text{ особенная, полагается } D=0,\\ 2 — \text{определитель слишком велик},\\ 3 — \text{определитель слишком мал}.\end{cases}$$

12.3.10. Решение систем линейных алгебраических уравнений. Программы этого пункта вычисляют приближенные решения системы линейных уравнений

$$AX=B,$$

где матрица A — матрица системы. Матрица B образована набором векторов правых частей систем, в простейшем случае B есть один вектор b, матрица X образуется из решений x систем

$$Ax=b$$

с правыми частями, входящими в B, в простейшем случае X есть один вектор x.

Алгоритмы основаны на различных вариантах метода исключения Гаусса.

A9A0: *решение вещественной системы линейных уравнений с симметричной положительно определенной матрицей*

SUBROUTINE A9A0(A,K,B,L,N,M,X,J,W,W1,L1,I)

Параметры входные:

A — вещественный двумерный массив, размерность (K, P), где $P \geqslant N$, содержит матрицу A, используется только верхняя треугольная часть,

K — целое, число строк массива A, описанное в вызывающей программе, $K \geqslant N$; в простом варианте положить $P=K=N$,

B — вещественный двумерный массив, размерность (L, P), где $P \geqslant M$, содержит элементы M векторов b, записанные по столбцам,

L — целое, число строк массива B, описанное в вызывающей программе, $L \geqslant N$; в простом варианте положить $L=N$,

N — целое, порядок матрицы A,

M — целое, число векторов B,

J — целое, число строк массива X, описанное в вызывающей программе, $J \geqslant N$; в простом варианте положить $J=N$,

W — вещественный одномерный массив, размерность N,

L1 — целое, число строк массива W1, описанное в вызывающей программе, L1 $\geqslant$ N,

I — индекс ошибки, до обращения положить I=0;

выходные:

X — вещественный двумерный массив, размерность (J, P), где $P \geqslant M$; содержит M векторов решений системы,

W1 — вещественный двумерный массив, размерность $L \times P$, P ⩾ M; содержит матрицу невязок $B - AX$;

$$I = \begin{cases} 0 — \text{нет ошибок,} \\ 1 — \text{матрица } A \text{ не положительно определена,} \\ 2 — \text{матрица } A \text{ плохо обусловлена.} \end{cases}$$

A9A1: *решение вещественной системы линейных уравнений*

SUBROUTINE A9A1(A,K,B,L,N,M,X,J,W,I)

Параметры входные:

A — вещественный двумерный массив, размерность (K, P), где $P \geqslant N$; содержит матрицу A,

K — целое, число строк A, описанное в вызывающей программе, $K \geqslant N$; в простом варианте положить $P = K = N$,

B — вещественный двумерный массив, размерность (L, P), где $P \geqslant M$; содержит элементы M векторов b, записанных по столбцам,

L — целое, число строк B, описанное в вызывающей программе, $L \geqslant N$; в простом варианте положить $L = N$,

N — целое, порядок матрицы A,

M — целое, число векторов B,

J — целое, число строк массива X, описанное в вызывающей программе, $J \geqslant N$; в простом варианте положить $J = N$,

W — вещественный одномерный массив, размерность N,

I — индекс ошибок; до обращения положить I = 0;

выходные:

X — вещественный двумерный массив, размерность $J \times P$, где $P \geqslant M$; содержит M векторов решений системы;

$$I = \begin{cases} 0 — \text{нет ошибок,} \\ 1 — \text{матрица } A — \text{особенная.} \end{cases}$$

12.3.11. Вычисление собственных значений и векторов. В настоящем пункте представлены программы, определяющие собственные значения матрицы A, т. е. ненулевые λ, удовлетворяющие равенству $Ax = \lambda x$, и соответствующие λ — векторы x — собственные векторы матрицы A. В основе алгоритмов лежит: приведение исходной марицы к трехдиагональной (симметричные матрицы) или к почти треугольной (в общем случае) и применение QL- или QR-алгоритмов.

B0A0: *вычисление собственных значений вещественной симметричной матрицы*

SUBROUTINE B0A0(A,K,N,G,W,I)

Параметры входные:

A — вещественный двумерный массив, размерность (K, P), где $P \geqslant N$; содержит матрицу A, для вычислений нужна нижняя треугольная часть,

K — целое, число строк A, описанное в вызывающей программе, $K \geqslant N$; в простом варианте положить $P = K = N$,

N — целое, порядок матрицы A,

W — вещественный одномерный массив, размерность N, рабочий массив,

I — индекс ошибки, до обращения положить I = 0;

выходные:

G — вещественный одномерный массив, размерность N, содержит $\lambda_1, \lambda_2, \ldots, \lambda_n$ в порядке возрастания;

I = { 0 — нет ошибок,
1 — алгоритм медленно сходится из-за близости собственных значений.

B0A1: *вычисление собственных значений и собственных векторов вещественной симметричной матрицы*

SUBROUTINE B0A1(A,K,N,G,X,L,W,I)

Параметры входные:

A — вещественный двумерный массив, размерность (K, P), где $P \geqslant N$; содержит матрицу A, для вычислений нужна нижняя треугольная часть,

K — целое, число строк A, описанное в вызывающей программе, $K \geqslant N$; в простом варианте положить $P = K = N$,

N — целое, порядок матрицы A,

L — целое, число строк массива X; описанное в вызывающей программе, $L \geqslant N$,

W — вещественный одномерный массив, размерность N, рабочий массив,

I — индекс ошибки, до обращения положить I = 0;

выходные:

G — вещественный одномерный массив, размерность N, содержит $\lambda_1, \lambda_2, \ldots, \lambda_n$ в порядке возрастания,

X — вещественный двумерный массив, размерность (L, S), где $S \geqslant N$; содержит нормированные собственные вектора

(в евклидовой норме), расположенные по столбцам с соответствием

X(J,I)↔G(I), J=1, 2, ..., N;

I = { 0 — нет ошибок,
1 — алгоритм медленно сходится из-за близости собственных значений.

B0A2: *вычисление собственных значений вещественной матрицы*

SUBROUTINE B0A2(A,K,N,G1,G2,J,I)

Параметры входные:

A — вещественный двумерный массив, размерность (K, P), где $P \geqslant N$; содержит матрицу A,

K — целое, число строк A, описанное в вызывающей программе, $K \geqslant N$; в простом варианте положить $P=K=N$,

N — целое, порядок матрицы A,

J — целый одномерный массив, размерность N, рабочий массив,

I — индекс ошибки; до обращения положить I=0;

выходные:

G1 — вещественный одномерный массив, размерность N; содержит вещественные части $\mathrm{Re}\,\lambda_i$, $1 \leqslant i \leqslant N$, собственных значений,

G2 — вещественный одномерный массив, размерность N; содержит мнимые части $\mathrm{Im}\,\lambda_i$, $1 \leqslant i \leqslant N$, собственных значений,

I = { 0 — нет ошибок,
1 — алгоритм медленно сходится из-за близости собственных значений.

B0A3: *вычисление собственных значений и собственных векторов вещественной матрицы*

SUBROUTINE B0A3(A,K,N,G1,G2,X1,L1,X2,L2,J,I)

Параметры входные:

A — вещественный двумерный массив, размерность (K, P), где $P \geqslant N$; содержит матрицу A,

K — целое, описанное в вызывающей программе число строк A, $K \geqslant N$, в простом варианте положить $P=K=N$,

N — целое, порядок матрицы A,

L1 — целое, число строк массива X1, описанное в вызывающей программе, $L1 \geqslant N$,

L2 — целое, число строк массива X2, описанное в вызывающей программе, $L2 \geqslant N$,

J — целый одномерный массив, размерность N, рабочий массив,

I — индекс ошибки, до обращения положить I=0;

выходные:

G1 — вещественный массив, размерность N; содержит вещественные части $\mathrm{Re}\,\lambda_i$, $1 \leqslant i \leqslant N$, собственных значений

G2 — вещественный массив, размерность N; содержит мнимые части $\mathrm{Im}\,\lambda_i$, $1 \leqslant i \leqslant N$, собственных значений,

X1 — вещественный двумерный массив, размерность (L1, S), где $S \geqslant N$; содержит вещественные части собственных векторов. Соответствие X1(J,I)↔G(I)+iG2(I),

X2 — вещественный двумерный массив, размерность (L2, S), где S⩾N; содержит мнимые части собственных векторов. Соответствие X2(J,I)↔G(I)+iG2(I);

I = { 0 — нет ошибок,
1 — алгоритм медленно сходится из-за близости собственных значений.

12.3.12. Линейная оптимизация. Рассматривается каноническая задача линейной оптимизации: найти вектор $x=(x_1, \ldots, x_n)$, дающий минимум линейной функции

$$L(x)=(c, x)=\sum_{i=1}^{n} c_i x_i$$

при ограничениях

$$Ax=b, \quad x_i \geqslant 0, \quad 1 \leqslant i \leqslant n;$$

матрица $A=(A_{i,j})$, $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n$, вектор $b=(b_1, \ldots, b_m)$.

B1A0: *решение задачи линейной оптимизации симплекс-методом*

SUBROUTINE B1A0(W,M,U,N,X,K,I,E,V,A,B,C)

Параметры входные:

W — вещественный одномерный массив, размерность (M, N), рабочий массив,

M — целое, равное $m+2$, m — число строк массива A,

U — вещественный одномерный массив, размерность (M, M), рабочий массив,

N — целое, равное n, n — число неизвестных x_i,

E — абсолютная точность выполнения неравенств $x_i \geqslant 0$,

V — вещественный одномерный массив, размерность M, рабочий массив,

A — вещественный двумерный массив, размерность $((M-2), N)$, содержит матрицу A,

B — вещественный одномерный массив, размерность $(M-2)$, содержит вектор b,

C — вещественный одномерный массив, размерность N, содержит вектор $c=(c_1, ..., c_n)$

выходные:

X — вещественный одномерный массив, размерность M, содержит оптимальное решение,

K — целый одномерный массив, размерность $M-2$, содержит номера переменных в массиве X.

Если

$K(1)=K_1$, $K(2)=K_2$, ..., $K(M-2)=K_m$,

$X(1)=\alpha_1$, $X(2)=\alpha_2$, ..., $X(M-2)=\alpha_m$,

то

$x_{K_1}=\alpha_1$, $x_{K_2}=\alpha_2$, ..., $x_{K_m}=\alpha_m$.

Оптимальное значение L содержится в $X(M-1)$. Если $K_i>10^6$, то номер переменной 10^6-K_i.

Значения переменных x_i, номера которых отсутствуют в массиве K, равны нулю.

I — индекс ошибки;

$$I=\begin{cases}1 — \text{найдено оптимальное решение,}\\ 2 — \text{задача не имеет решения,}\\ 3 — \text{функция } L(x) \text{ не ограничена.}\end{cases}$$

12.3.13. Решение линейных интегральных уравнений. Рассматривается интегральное уравнение Фредгольма второго рода

$$y(x)=\lambda\int_a^b G(x, s)y(s)\,ds+f(s),\quad a\leqslant x\leqslant b.$$

Решение $y(x)$ при фиксированном λ ищется в виде отрезка ряда Чебышева одной из форм

$$y(x)=\begin{cases}0{,}5c_1+\sum\limits_{i=2}^{N} c_i T_{i-1}(x),\\ 0{,}5c_1+\sum\limits_{i=2}^{N} c_i T_{2i-2}(x), \text{ четная } y(x),\\ \sum\limits_{i=1}^{N} c_i T_{2i-1}(x), \text{ нечетная } y(x)\end{cases}$$

(на приведенном интервале $-1\leqslant x\leqslant +1$). Исходное уравнение сводится к системе линейных алгебраических уравнений относительно вектора $c=(c_1, c_2, ..., c_N)$. Факт сходимости выясняется путем сравнения решений систем с несколькими разными N и сравнения результатов вычислений с дополнительными данными о точном решении. Если известно, что решение $y(x)$ — четная функция относительно середины отрезка $[a, b]$, то ищется разложение по

полиномам Чебышева четной степени, если — нечетная, то — по полиномам Чебышева нечетной степени.

B2A0: *решение линейного интегрального уравнения Фредгольма второго рода*

SUBROUTINE B2A0(G,F,P,A,B,R,D,N,W,V,Z,K,L,Y,C,I)

Параметры входные:

G — внешняя вещественная функция, вычисляющая по вещественным аргументам x, s значение ядра $G(x, s)$,
F — внешняя вещественная функция, вычисляющая по вещественному аргументу x значение $f(x)$,
P — вещественное, значение λ,
A — нижний предел интегрирования a,
B — верхний предел интегрирования b,
R — логическая переменная. Значение
.TRUE.— рассматривается четное или нечетное решение в зависимости от D,
.FALSE.— рассматривается общий случай,
D — логическая переменная. Значение
.TRUE.— четное решение $y(x)$,
.FALSE.— нечетное решение $y(x)$,
N — целое, длина отрезка ряда,
W — вещественный двумерный массив, размерность (K, M), $M \geqslant K$, рабочий массив,
V — вещественный двумерный массив, размерность (K, N), $N \geqslant 1$, рабочий массив,
Z — вещественный двумерный массив, размерность $(2, P)$, $P \geqslant L$, рабочий массив,
K — целое, описанное в вызывающей программе число строк массива W, V, $K \geqslant N$;
L — целое, $L = 2N + 1$,
I — индекс ошибки, до обращения положить I=0;

выходные:

Y — вещественный одномерный массив, размерность N, содержит приближенные значения $y(x_i)$ в первых N из $M+1$-й точки

$$x_i = \frac{1}{2}\left(a + b + (b - a)\cos\left(\frac{(i-1)\pi}{M}\right)\right), \quad 1 \leqslant i \leqslant M + 1,$$

если R=.TRUE. и D=.TRUE., M=2N
если R=.TRUE. и D=.FALSE., M=2N−1
если R=.FALSE., M=N−1

C — вещественный одномерный массив, размерность N, коэффициенты c_i $1 \leqslant i \leqslant N$;

I = 0 — нет ошибок,
1 — $A \geqslant B$,
2 — P близко к собственному значению интегрального уравнения.

12.3.14. Корни полиномов

B3A0: *определение корней полинома с вещественными коэффициентами*

Программа вычисляет корни полинома

$$P_N(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_N x^N.$$

SUBROUTINE B3A0(A,W,N,X,Y,I)

Параметры входные:

A — вещественный одномерный массив, размерность $N+1$, содержащий коэффициенты полинома, $A(i+1) = a_i$, $0 \leqslant i \leqslant N$,
W — вещественный одномерный массив, размерность $N+1$, рабочий массив,
N — порядок полинома

выходные:

X — вещественный одномерный массив, размерность N, содержит действительные части корней,
Y — вещественный одномерный массив, размерность N, содержит мнимые части корней,
I — индекс ошибки;

I = 0 — нет ошибок,
1 — $N < 1$,
2 — $N > 36$,
3 — корень не определяется за допустимое число итераций,
4 — $a_N = 0$.

12.3.15. Корни нелинейных уравнений

B4A0: *поиск корня непрерывной функции одной переменной без вычисления производной*

Программа вычисляет приближение к корню уравнения

$$f(x) = 0$$

по заданному интервалу $[a, b]$, на котором локализован единственный корень, т. е. $f(a) \cdot f(b) \leqslant 0$.

В основе алгоритма метод бисекции.

SUBROUTINE B4A0(X,R,F,A,B,E,N,I)

Параметры входные:

F — имя внешней вещественной функции для вычисления $f(x)$,

A — вещественное, левый конец интервала неопределенности корня, равен a,
B — вещественное, правый конец интервала неопределенности корня, равен b,
E — вещественное, абсолютная точность определения корня ε,
N — целое, максимальное значение числа итераций;

выходные:

X — вещественное, приближенное значение корня,
R — вещественное, значение $f(x)$,
I — индекс ошибки,

$$I=\begin{cases}0 — \text{нет ошибок},\\ 1 — \text{точность } \varepsilon \text{ не достигнута за } N \text{ итераций},\\ 2 — \text{нарушено условие } f(a)\cdot f(b)\leqslant 0.\end{cases}$$

B4A1: *поиск корня функции одной переменной методом Ньютона*

Программа реализует метод Ньютона

$$x_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)},\quad k=0,\ 1,\ 2,\ \ldots$$

по заданному x_0 с условием прекращения итераций по заданной точности ε. Процесс прекращается, если

$$\delta\leqslant\varepsilon \text{ и } |f(x_{k+1})|\leqslant 10^2\varepsilon,$$

где

$$\delta=\begin{cases}|x_{k+1}-x_k|/|x_{k+1}|, \text{ если } |x_{k+1}|>1,\\ |x_{k+1}-x_k|, \text{ если } |x_{k+1}|\leqslant 1.\end{cases}$$

SUBROUTINE B4A1(X,R,D,F,X0,E,N,I)

Параметры входные:

F — имя внешней подпрограмы, вычисляющей по заданному вещественному аргументу x вещественное значение R и производной D, список ее параметров (X, R, D),
X0 — вещественное, начальное приближение к корню,
E — вещественное, заданная точность ε,
N — целое, максимальное число итераций;

выходные:

X — вещественное, приближенное значение корня,
R — вещественное, значение $f(x)$,
D — вещественное, значение $f'(x)$,
I — индекс ошибок;

$$I=\begin{cases}0 — \text{нет ошибок},\\ 1 — \text{точность } \varepsilon \text{ не достигнута за } N \text{ итераций},\\ 2 — \text{значение } f'(x) \text{ близко к нулю}.\end{cases}$$

B4A2: *решение системы нелинейных уравнений*

В основе алгоритма — метод Ньютона для векторного уравнения $f(x)=0$

$$f_i(x_1, ..., x_n)=0, \quad 1 \leqslant i \leqslant n.$$

По заданному начальному приближению $x^{(0)}=(x_1^{(0)}, ..., x_n^{(0)})$ к корню определяются последовательные приближения $x^{(k)}=(x_1^{(k)}, ..., x_n^{(k)})$, $k=1, 2, ...$, к корню. Процесс прекращается по заданной точности ε, если

$$\| x_k - x_{k+1} \| \leqslant \varepsilon \| x_{k+1} \|$$

в евклидовой норме.

Описание

SUBROUTINE B4A2(R,N,X,F,E,W,K,I)

Параметры входные:

R — имя внешней подпрограммы, вычисляющей значения f_i по вектору x, список параметров (N, X, F, J), где
N — целое — число уравнений,
X — вещественный одномерный массив, размерность N,
F — вещественный одномерный массив, размерность N, должен содержать f_i,
J — индекс ошибки, если положить $J=-1$ вычисления прерываются, это значение присваивается I,
N — то же, что и в R,
X — то же, что и в R, перед обращением присвоить значения $(x_1^{(0)}, ..., x_n^{(0)})$,
F — то же, что и в R,
E — вещественное, значение ε,
W — вещественный одномерный массив, размерность K, рабочий массив,
K — целое, размерность массива W, $K \geqslant N(3N+3)/2$,
I — целое, индекс ошибки, положить до обращения $I=0$;

выходные:

X — содержит приближенное значение к решению системы уравнений, вектор $(x_1, ..., x_n)$,
F — содержит значение $f_i(x_1, ..., x_n)$, $1 \leqslant i \leqslant n$;

$$I=\begin{cases} 0 & \text{— нет ошибок,} \\ -1 & \text{— прерывание в подпрограмме } R, \\ 1 & \text{— } N \leqslant 0,\ E<0,\ K<N(3N+3)/2, \\ 2 & \text{— выполнено } 200(N+1) \text{ итераций,} \\ 3 & \text{— слишком мало } \varepsilon. \end{cases}$$

12.3.16. Нелинейная оптимизация

B5A0: *минимизация функции одной переменной*

Программа минимизирует функцию $f(x)$ в заданном интервале $[a, b]$ (определяет локальный минимум).

SUBROUTINE B5A0(F,E,E1,A,B,N,X,R,I)

Параметры входные:

F — имя внешней подпрограммы, вычисляющей значение $f(x)$. Список параметров подпрограммы (X1, F1), где $F1=f(X1)$,

E — вещественное, относительная точность определения точки минимума,

E1 — вещественное, абсолютная точность определения точки минимума,

A — вещественное, значение левого конца интервала, равно a,

B — вещественное, значение правого конца интервала, равно b,

N — максимальное число итераций поиска минимума,

I — индекс ошибки, до обращения положить I=0;

выходные:

X — вещественное, приближенное значение точки минимума с точностью $E|X|+E1$,

R — вещественное, значение $f(X)$,

I = 0 — нет ошибок,
I = 1 — $A+E\geqslant B$, $N<3$,
I = 2 — число итераций для достижения заданной точности $>N$.

B5A1: *минимизация функции нескольких переменных*

Программа определяет локальный минимум функции $f(x)=f(x_1, ..., x_n)$ по заданному начальному приближению $x^0=(x_1^0, ..., x_n^0)$. В основе алгоритма — градиентный метод.

SUBROUTINE B5A1(F,N,X,R,G,S,E,K,I,W)

Параметры входные:

F — имя внешней подпрограммы, вычисляющей значение $f(x)$ и grad $f(x)$. Список параметров подпрограммы (N, X, Y, G),

N — целое, размерность вещественного массива X,

X — вещественный одномерный массив, размерность N,

Y — вещественное, значение $f(x)$,

G — вещественный одномерный массив, размерность N, содержит вектор градиента,

N — целое, размерность вектора x,

X — вещественный одномерный массив, размерность N, до обращения элементам массива присвоить значения $(x_1^0, ..., x_n^0)$,

S — вещественное, оценка минимального значения функции,

E — вещественное, абсолютная погрешность в евклидовой норме,

K — целое, максимальное число итераций,
W — вещественный одномерный массив, размерность $2N$, рабочий массив;

выходные:

X — содержит вектор x, доставляющий минимум $f(x)$,
R — вещественное, содержит значение $f(x)$,
G — вещественный одномерный массив, размерность N, содержит компоненты вектора градиента,
I — индекс ошибки;

$$I=\begin{cases}0 & \text{— нет ошибок,}\\ 1 & \text{— точность не достигается за } K \text{ итераций,}\\ -1 & \text{— ошибка в вычислениях } \operatorname{grad} f(x),\\ 2 & \text{— нет минимума.}\end{cases}$$

12.3.17. Решение обыкновенных дифференциальных уравнений; задача Коши

B6A0: *интегрирование системы обыкновенных дифференциальных уравнений методом Рунге — Кутта*

Программа определяет решение системы уравнений

$$\frac{dy_i}{dx}=f_i(x, y_1, \ldots, y_n),\quad a\leqslant x\leqslant b,$$

удовлетворяющее начальным условиям $y_i(a)=y_i^0$, $1\leqslant i\leqslant n$, с выбором шага интегрирования по заданной точности методом Рунге — Кутта. Для вывода значений $y_i(x)$ в точках интервала $[a, b]$, следует составить программу вывода (см. ниже), в этих точках $y_i(x)$ находится интерполированием в программе B6A0.

SUBROUTINE B6A0(X,B,N,Y,E,J,F,O,W,I)

Параметры входные:

X — вещественное, значение независимого переменного x, перед обращением присвоить $X=a$,
B — вещественное, значение правого конца интервала интегрирования, равно b,
N — целое, число уравнений системы, равно n,
Y — вещественный одномерный массив, размерность N, содержит значения $y_1(x)$, $y_2(x)$, ..., $y_n(x)$ перед обращением присвоить элементам массива начальные значения $(y_1^0, y_2^0, \ldots, y_n^0)$,
E — вещественное, значение задаваемой погрешности ε,
J — целое, индекс контроля погрешности, до обращения положить

$$J=\begin{cases}0 & \text{— если контроль по условию } \delta\leqslant\varepsilon \max\left[1, |y_1(x)|, \ldots, |y_n(x)|\right],\\ 1 & \text{— если контроль по условию } \delta\leqslant\varepsilon,\\ & \text{где } \delta \text{ — вычисленная оценка погрешности на шаге,}\end{cases}$$

F — имя внешней подпрограммы, вычисляющей правые части системы уравнений $f_i(x, y)$,
Список параметров (X, Y, R):
X — то же, что в B6A0,
Y — то же, что в B6A0,
R — вещественный одномерный массив, размерность N, на выходе должен содержать значения $(f_1(X, Y), f_2(X, Y), ..., f_n((X, Y))$,
O — имя внешней подпрограммы вывода результатов вычислений. Список параметров подпрограммы (X1, Y1),
X1 — вещественное, перед обращением к подпрограмме O содержит текущее значение x; на выходе из подпрограммы X1 должен содержать следующее значение x, при котором будет вызов O,
Y1 — вещественный одномерный массив, размерность N, содержит y_i(X1), $1 \leqslant i \leqslant N$,
W — вещественный двумерный массив, размерность $(N, 7)$, рабочий массив,
I — индекс ошибки, до обращения положить I=0;

выходные:

Y — содержит значения $(y_1(b), y_2(b), ..., y_n(b))$;

$$\mathrm{I}=\begin{cases}0 & \text{— нет ошибок,}\\ 1 & \text{— } E \leqslant 0,\ N \leqslant 0,\ J \neq 0 \text{ или } 1,\\ >1 & \text{— заданная точность не может быть обеспечена.}\end{cases}$$

B6A1: *интегрирование системы жестких обыкновенных дифференциальных уравнений*

Программа служит для решения системы жестких уравнений

$$\frac{dy_i}{dx}=f_i(x, y_1, ..., y_n), \quad a \leqslant x \leqslant b,$$

с начальными условиями

$$y_i(a)=y_i^0, \quad 1 \leqslant i \leqslant n.$$

SUBROUTINE B6A1(X,B,N,Y,E,J,F,M,P,O,W,K,I)

Параметры, кроме W, одноименные с параметрами программы B6A0, имеют тот же смысл. Поэтому опишем только дополнительные параметры входные:

M — целое, индекс формы вычисления якобиана
$\frac{\partial f_i}{\partial y_i}(x, y_1, ..., y_n)$, $1 \leqslant i, j \leqslant n$;

$$M = \begin{cases} 1 — \text{по внешней подпрограмме } P, \\ 0 — \text{внутри B6A1}, \end{cases}$$

P — имя внешней подпрограммы, вычисляющей якобиан по значениям $(x, y_1, ..., y_n)$. Список параметров подпрограммы (X, Y, Z),
Параметры X, Y, имеют тот же смысл, что и в B6A0

Z — вещественный, двумерный массив, размерность (N, N), на выходе должен содержать значения

$$\frac{\partial f_i}{\partial y_j}(x, y), \quad 1 \leqslant i, \ j \leqslant n,$$

W — вещественный, двумерный массив, размерность (N, K), рабочий массив,

K — целое число столбцов массива W, необходимо $K \geqslant N + 18$.

12.3.18. Решение обыкновенных дифференциальных уравнений; краевая задача

B7A0: *решение краевой задачи для системы линейных обыкновенных дифференциальных уравнений*

Программа определяет приближенное решение системы уравнений

$$\frac{dy_i}{dx} = \sum_{j=1}^{n} Q_{i,j}(x) y_i + f_i(x), \quad a < x < b, \quad 1 \leqslant i \leqslant n,$$

с краевыми условиями

$$\sum_{j=1}^{n} G_{i,j} y_j(a) + \sum_{j=1}^{n} D_{i,j} y_j(b) = c_i, \quad 1 \leqslant i \leqslant n,$$

где $Q_{i,j}(x)$ — матрица, $f_i(x)$ — вектор, зависящие от x, матрицы $G_{i,j}$, $D_{i,j}$, вектор c_i — постоянные, $1 \leqslant i, j \leqslant n$. В основе алгоритма — метод конечных разностей. Интервал $[a, b]$ разбивается узлами x_m, $1 \leqslant m \leqslant n_1$. Приближенное решение определяется в узлах $x_m (x_1 = a, x_{n_1} = b, x_m < x_{m+1})$ по заданной точности ε. Если выполняется условие

$$\max_{1 \leqslant i \leqslant n} \max_{1 \leqslant m \leqslant n_1} |y_i(x_m) - y_{i,m}| \leqslant \varepsilon,$$

то процесс вычислений должен прекратиться, здесь $y_{i,m}$ — решение разностных уравнений, $y_i(x_m)$ — значение точного решения в узлах.

SUBROUTINE B7A0(A,B,N,E,Q,F,G,D,C,M,X,Y,N1,W,L,J,K,I)

Параметры входные:

A — вещественное, значение левого конца интервала интегрирования, равен a,

B — вещественное, значение правого конца интервала интегрирования, равен b,

N — целое, число уравнений в системе, равно n,

E — вещественное, абсолютная точность, равно ε,

Q — имя внешней подпрограммы, вычисляющей матрицу $Q_{i,j}(x)$ в точке x. Ее параметры (X, Z),

X — вещественное, значение x,

Z — вещественный двумерный массив, размерность (N, N), на выходе должен содержать элементы массива $Q_{i,j}(x)$,

$$Z(I, J) = Q_{i,j}(x),$$

F — имя внешней подпрограммы, вычисляющей вектор $f_i(x)$ в точке x. Ее параметры (X, R),

X — вещественное, значение x,

R — вещественный одномерный массив, размерность N, на выходе должен содержать элементы массива $f_i(x)$, $R(I) = f_i(x)$,

G — вещественный двумерный массив, размерность (N, N), содержит элементы матрицы $G_{i,j}$,

D — вещественный двумерный массив, размерность (N, N), содержит элементы матрицы $D_{i,j}$,

C — вещественный одномерный массив, размерность N, содержит элементы вектора c_i,

M — целое, максимально допустимое число узлов, должно быть $M \geqslant 32$,

X — вещественный одномерный массив, размерность M, содержит начальные узлы

$$A = X(1) < X(2) < \ldots < X(N) = B,$$

N1 — целое, на входе содержит начальное число узлов, $N1 \geqslant 4$,

W — вещественный одномерный массив, размерность L, рабочий массив,

L — целое, размерность W, $L \geqslant M(3N^2 + 5N + 7) + 3N^2 + 5N$,

J — целый одномерный массив, размерность K, рабочий массив,

K — целое, размерность J, $J \geqslant M(2N + 1) + N$,

I — индекс ошибки, на входе присвоить I = 110

выходные:

X — содержит значения узлов сетки, обеспечивающих заданную точность,

Y — вещественный двумерный массив, размерность (N, M) содержит массив $y_{i,j}$ — приближенное решение в узлах x_j, $Y(I, J)=y_{i,j}$, $1 \leqslant I \leqslant N$, $1 \leqslant J \leqslant$ N1,

N1 — содержит число узлов сетки, обеспечивающих заданную точность,

I = { 0 — нет ошибок,
1 — входные параметры лежат вне допустимых пределов,
2 — одна из матриц, G или D, — нулевая,
3 — точность ε не может быть достигнута.

B7A1: *решение краевой задачи для системы нелинейных обыкновенных дифференциальных уравнений*

Программа определяет приближенное решение системы уравнений

$$\frac{dy_i}{dx}=f_i(x, y_1, \ldots, y_n), \quad a<x,$$

с условиями

$$y_e(a)=y_e^0, \quad 1 \leqslant e \leqslant k, \quad k<n,$$
$$y_p(b)=y_p^1, \quad k<p \leqslant n,$$

y_e^0, y_p^1 — заданные числа. В основе алгоритма — метод конечных разностей с дальнейшим решением системы нелинейных уравнений методом Ньютона. Для начального приближения необходимо задать свободные компоненты вектора y в точках a и b (грубое приближение), т. е.

$$y(a)=(y_1^0, \ldots, y_k^0, y_{k+1}^0, \ldots, y_n^0),$$
$$y(b)=(y_1^1, \ldots, y_k^1, y_{k+1}^1, \ldots, y_n^1),$$

вычисления заканчиваются при достижении точности ε так же, как в B7A0.

SUBROUTINE B7A1(U,V,N,A,B,E,F,M,X,Y,N1,W,L,J,K,I)

Параметры входные:

U — вещественный двумерный массив, размерность $(N, 2)$, перед обращением следует присвоить

$$U(\mathrm{I}, 1)=y_i^0, \quad U(\mathrm{I}, 2)=y_i^1, \quad 1 \leqslant \mathrm{I} \leqslant N,$$

V — вещественный двумерный массив, размерность $(N, 2)$, перед обращением следует присвоить

V(I, 1)=0., 1⩽I⩽K, V(I, 1)=1., K⩽I+1⩽N
V(I, 2)=1., 1⩽I⩽K, V(I, 2)=0., K⩽I+1⩽N

N — целое, число уравнений, равно n,

A — вещественное, значение левого конца интервала интегрирования, равно a,

B — вещественное, значение правого конца интервала интегрирования, равно b,

E — вещественное, абсолютная точность, равно ε,

F — имя внешней подпрограммы, вычисляющей вектор $f_i(x, y_1, ..., y_n)$, ее параметры (X, Y, R),

X — вещественное, значение x,

Y — вещественный одномерный массив, размерность N, содержит вектор $(y_1, ..., y_n)$,

R — вещественный одномерный массив, размерность N, на выходе должен содержать вектор $(f_1(x, y), ..., f_n(x, y))$

M — целое, максимально допустимое число узлов, $M \geqslant 32$,

X — вещественный одномерный массив, размерность M, содержит начальные узлы

$$A = X(1) < X(2) < ... < X(N1) = B,$$

N1 — целое, содержит начальное число узлов $N1 \geqslant 4$,

W — вещественный одномерный массив, размерность L, рабочий массив,

L — целое, размерность W, $L \geqslant M(3N^2+6N+7)+4N^2+4N$,

J — целый одномерный массив, размерность K, рабочий массив,

K — целое, размерность J, $K \geqslant M(2N+1)+N^2+4N+2$,

I — индекс ошибки, до обращения присвоить I = 110

выходные:

X — содержит значения узлов сетки, обеспечивающих заданную точность

$$A = X(1) < X(2) < ... < X(N1) = B,$$

Y — вещественный двумерный массив, размерность (N, M), содержит массив $y_{i,j}$ — приближенное решение в узлах x_j,

$$Y(I, J) = y_{i,j}, \quad 1 \leqslant I \leqslant N, \quad 1 \leqslant J \leqslant N,$$

N1 — содержит число узлов сетки, обеспечивающих заданную точность;

I =
- 0 — нет ошибки,
- 1 — входные параметры лежат вне допустимых пределов,
- 2 — нет сходимости метода Ньютона,
- 3 — слишком мало ε,
- 4 — мало M.

12.3.19. Решение уравнений с частными производными

B8A0: *решение эллиптического уравнения в прямоугольнике разностным методом*

Программа находит решение разностной системы уравнений

$$a_{i,j}u_{i,j-1} + b_{i,j}u_{i-1,j} + c_{i,j}u_{i,j} + d_{i,j}u_{i+1,j} + e_{i,j}u_{i,j+1} = f_{i,j},$$

которая получается при решении эллиптических уравнений (Лапласа, Пуассона и т. д.) методом сеток. Будем считать, что краевая задача сведена к решению разностной системы уравнений

$$c_{i,j} \neq 0, \quad 1 \leqslant i \leqslant n_1, \quad 1 \leqslant j \leqslant n_2$$

Метод решения — итерационный, для начала вычислений следует задать начальное приближение $u_{i,j}$, $1 \leqslant i \leqslant n_1$, $1 \leqslant j \leqslant n_2$.

```
SUBROUTINE B8A0(N1,N2,N,A,B,C,D,E,F,U,G,I1,I2,I3,
                N3,I4,I5,E1,E2,E3,E4,W1,W2,W3,I)
```

Параметры входные:

N1 — целое, число узлов по первой координате,

N2 — целое, число узлов по второй координате,

N — целое, число строк в массивах A,B,C,D,E,F,V,W1, W2,W3; $N \geqslant N1$,

A — вещественный двумерный массив, размерность (N, P), $P \geqslant N2$ на входе должен содержать массив $a_{i,j}$, причем $A(I, 1) = 0$,

B — вещественный двумерный массив, размерность (N, P), $P \geqslant N2$, на входе должен содержать массив $b_{i,j}$, причем $B(1, J) = 0$,

C — вещественный двумерный массив, размерность (N, P), $P \geqslant N2$, на входе должен содержать массив $c_{i,j}$,

D — вещественный двумерный массив, размерность (N, P), $P \geqslant N2$, на входе должен содержать массив $d_{i,j}$, причем $D(N1, J) = 0$.

E — вещественный двумерный массив, размерность (N, P), $P \geqslant N2$, на входе должен содержать массив $e_{i,j}$, причем $E(I, N2) = 0$,

F — вещественный двумерный массив, размерность (N, P), $P \geqslant N2$, на входе содержит массив $f_{i,j}$,

U — вещественный двумерный массив, размерность (N, P), $P \geqslant N2$, на входе должен содержать начальное приближение $u_{i,j}^{(0)}$,

G — вещественное, коэффициент ускорения итераций, обычно полагают $G = 1$, если сходимость медленная — положить $G = 0.1$,

I1 — целое, максимальное число итераций,

I2 — целое, перед входом присвоить $I2 = 0$,

N3 — целое, перед входом положить $N3 = 0$; если $N3 \neq 0$, то это признак неединственности решения разностных уравнений, такую ситуацию мы не рассматриваем,

I4, I5 — целое, если N3 = 0, то эти параметры не имеют значения,

E1 — вещественное, задаваемая точность выполнения алгебраических уравнений ε_1

$$\varepsilon_1 = \max_{i,j} |a_{i,j}u_{i,j-1} + \ldots + e_{i,j}u_{i,j+1} - f_{i,j}|/|c_{i,j}|,$$

E2 — вещественное, задаваемая точность изменения решения при переходе от k-й итерации к $(k+1)$-й

$$\varepsilon_2 = \max_{i,j} |u_{i,j}^{(k+1)} - u_{i,j}^{(k)}|$$

вычисления прекращаются при достижении точности E1 и E2.

W1, W2, W3 — вещественные двумерные массивы, размерность (N, P), P ⩾ N2, рабочие массивы,

I — индекс ошибки, до обращения положить I = 0;

выходные:

U — содержит решение $u_{i,j}$,

I2 — содержит общее число итераций при нескольких вызовах B8A0,

I3 — целое, содержит число итераций в данном обращении к B8A0,

E3 — вещественный одномерный массив, размерность I1, содержит $\varepsilon_1^{(k)}$ погрешность после k-й итерации,

$$E3(k) = \varepsilon_1^{(k)}, \quad k = 1, 2, \ldots, I1,$$

E4 — вещественный одномерный массив, размерность I1, содержит $\varepsilon_2^{(k)}$ — погрешность после k-й итерации

$$E4(k) = \varepsilon_2^{(k)}, \quad k = 1, 2, \ldots, I1;$$

$$I = \begin{cases} 0 & \text{— нет ошибок,} \\ 1\text{—}4 & \text{— параметры вне допустимых пределов,} \\ 5 & \text{— сходимость не достигается за I1 итераций.} \end{cases}$$

B8A1: *решение одномерного параболического уравнения методом прямых*

Программа интегрирует уравнение

$$\frac{\partial u}{\partial t} = \frac{1}{x^m}\frac{\partial}{\partial x}\left(x^m q(x, t, u)\frac{\partial u}{\partial x}\right) + f\left(x, t, u, \frac{\partial u}{\partial x}\right)$$

в области $\{t_0 \leqslant t \leqslant t_1,\ a \leqslant x \leqslant b\}$. Показатель $m = 0, 1, 2$ соответствует трем координатным системам: прямоугольной, цилиндрической, сферической соответственно. Ищется решение $u(x, t)$, удовлетворя-

ющее начальному условию $u(x, t_0)=u_0(x)$, $a\leqslant x\leqslant b$, и граничным

$$s_0(t)u(a, t)+g_0(t)\frac{\partial u}{\partial x}(a, t)=r_0(t, u),$$

$$s_1(t)u(b, t)+g_1(t)\frac{\partial u}{\partial x}(b, t)=r_1(t, u).$$

Исходное уравнение дискретизацией по пространственным переменным (метод прямых) сводится к системе обыкновенных дифференциальных уравнений, которая затем интегрируется по t с переменным шагом, определяемым заданной точностью. Сетка по x — равномерная.

Дополнительные ограничения:

1) $q(x, t, u)>0$;

2) если q, f разрывны по x, точки разрыва следует включать в узлы сетки по x;

3) если $m\neq 0$, то $a\geqslant 0$, если $a=0$, то граничное условие в этой точке $\dfrac{\partial u}{\partial x}=0$;

4) $s_i^2(t)+g_i^2(t)\neq 0$;

5) если $g_i(t)=0$, то $r_i(t, u)=r_i(t)$.

SUBROUTINE B8A1(M,A,B,T0,T1,U,N,E,W,K,L,I)

Параметры входные:

M — целое, определяет координатную систему, равно m,

A — вещественное, левый конец пространственного интервала, равно a,

B — вещественное, правый конец пространственного интервала, равно b,

T0 — вещественное, на входе должен содержать значение t_0,

T1 — вещественное, конечное значение t_1,

U — вещественный одномерный массив, размерность N на входе должен содержать массив — начальный временно́й слой $u_0(x_i)$

$$x_i=a+(i-1)h,\quad i=1, 2, \ldots, N,\quad h=(b-a)/(N-1)$$

N — целое, число узлов пространственной сетки, $N\geqslant 3$,

E — вещественное, задает точность интегрирования по времени t на одном шаге. Вычисленная погрешность ε_1 должна удовлетворять условию

$$\varepsilon_1\leqslant E\left(1+\max_I|U(I)|\right),$$

W — вещественный одномерный массив, размерность K, рабочий массив,

K — целое, размерность массива W, необходимо $K > 16N+40$,

L — целое, индекс типа интегрирования;

$$L=\begin{cases}0 — \text{интегрирование с началом в } t_0,\\ 1 — \text{интегрирование с момента выхода из B8A1, в этом случае следует перед вызовом B8A1 задать T1 и I,}\end{cases}$$

I — индекс ошибки, перед входом положить I=0;

выходные:

T0 — значение текущего t, если нет ошибок; если есть ошибка — значение t, при котором возникла ошибка,

T1 — значение t_1, если нет ошибок; если есть ошибка — значение t, при котором возникла ошибка,

U — значение временного слоя в момент t,

L — если нет ошибок, $L=1$;

$$I=\begin{cases}0 & — \text{нет ошибок,}\\ 1—3 & — \text{интегрирование по } t \text{ невозможно с заданной точностью,}\\ 4 & — \text{параметры выходят за допустимые пределы.}\end{cases}$$

Для применения программы B8A1 следует написать две подпрограммы, определяющие функции в дифференциальном уравнении и краевых условиях.

Вычисление значений функций f, q должна выполнять

SUBROUTINE PDEF(X,T,U,DU,F,Q)

Параметры входные:

X — вещественное, содержит x,

T — вещественное, содержит t,

U — вещественное, содержит решение $u(x, t)$,

DU — вещественное, содержит $\frac{\partial u}{\partial x}(x, t)$;

выходные:

F — вещественное, должен содержать $f\left(x, t, u, \frac{\partial u}{\partial x}\right)$,

Q — вещественное, должен содержать $q(x, t, u)$.

Вычисление значений функций $s_i(t)$, $g_i(t)$, $r_i(t, u)$ должна выполнять

SUBROUTINE BNDY(T,U,I,S,G,R)

Параметры входные:

T — вещественное, содержит t,

U — вещественное, содержит граничное значение $u(x, t)$ в момент t,

I — целое, индекс границы, на входе;

$$\mathrm{I}=\begin{cases}0 — \text{должны вычислять } s_0,\ g_0,\ r_0,\\ 1 — \text{должны вычислять } s_1,\ g_1,\ r_1;\end{cases}$$

выходные:

S — вещественное, должен содержать $s_0(t)$, если $\mathrm{I}=0$; либо $s_1(t)$, если $\mathrm{I}=1$,

G — вещественное, должен содержать $g_0(t)$, если $\mathrm{I}=0$; либо $g_1(t)$, если $\mathrm{I}=1$,

R — вещественное, должен содержать $r_0(t, u)$, если $\mathrm{I}=0$; либо $r_1(t, u)$, если $\mathrm{I}=1$

ЗАДАЧИ И УПРАЖНЕНИЯ

Задачи и упражнения разделены на три уровня в зависимости от необходимого уровня подготовки.

I уровень (минимальный) соответствует массовому обучению в группе. Задачи распределены на 32 задания, что может соответствовать 32 неделям двух семестров. Завершающим этапом является типовой расчет «Исследование переходного процесса и оптимизация системы управления». Это достаточно типичная задача инженерно-технического содержания.

II уровень (средний). Задачи распределены по главам.

III уровень (углубленный). Задачи распределены по главам. Для решения предлагаемых задач и упражнений необходима работа с дополнительной литературой.

I УРОВЕНЬ

Темы всех 32 заданий представлены в следующем списке.

Номер задания	Тема
1.	Начальные сведения об операционной системе.
2.	Арифметические выражения фортрана.
3.	Ввод — вывод.
4.	Операторы передачи управления, циклы.
5.	Массивы.
6.	Функция-подпрограмма.
7.	Подпрограмма.
8.	Логические выражения.
9.	Аппроксимация. Интерполяция.
10.	Аппроксимация. Метод наименьших квадратов.
11.	Численное интегрирование.
12.	
13.	Системы линейных уравнений.
14.	
15.	Решение нелинейных уравнений
16.	
17.	Нелинейная оптимизация.
18.	
19.	Обыкновенные дифференциальные уравнения. Задача Коши.
20.	

Продолжение табл.

Номер задания	Тема
21. 22.	ОДУ. Краевая задача. Метод прогонки.
23. 24.	Задача Дирихле для уравнения Пуассона.
25.	Смешанная задача для одномерного уравнения теплопроводности.
26.	Смешанная задача для двумерного уравнения теплопроводности.
27.	Постановка задачи типового расчета (ТР). Построение математической модели.
28.	Выбор численного метода. Блок-схема алгоритма ТР.
29. 30.	Программирование. Отладка программ ТР
31. 32.	Проведение вычислений на ЭВМ ТР.

Задание 1. 1) Представить клавиатуру дисплея. Отметить назначение основных клавиш.

2) Фортран-программа состоит из операторов

```
      K=10
      WRITE (5, 1)K
1     FORMAT(I2)
      END
```

написать процесс прохождения фортран-программы в ОС РВ с полными спецификациями файлов и с сокращенными.

3) Заменить в 2) букву K на L, удалить две средние строки программы.

4) Распечатать содержимое каталога. Вывести на терминал файл листинга. Удалить файлы типа OBJ, LST, TSK.

5) Как войти в ОС РВ, выйти из ОС РВ?

6) Как вызвать файл для редактирования? Как выйти из редактора? В упражнениях 2)—6) пользователь задает имя файла, совпадающее с тремя первыми буквами его фамилии.

7) Привести примеры символов фортрана.

8) Привести примеры констант.

9) Привести примеры переменных.

Упражнения 2)—6) оформляются в виде дисплейной карты, а затем выполняются в дисплейном классе.

Замечание. Если работа на ЭВМ выполняется в операционной системе, отличной от ОС РВ, упражнения 2)—6) соответственно заменяются аналогичными действиями используемой ОС.

Задание 2. Написать программу вычисления значения функции $F(x)$ в точке x_0. Значение $x_0=0{,}5$ ввести оператором присваивания. Результат $y_0=f(x_0)$ вывести двумя операторами

```
      WRITE (6, 1)Y0
1     FORMAT(2X,E13.6)
```

Номер варианта	Функции $F(x)$
1.	$\sqrt[3]{e^{2,2x}}-\left\|\sin\frac{\pi x}{x+2/3}\right\|+1,7$
2.	$\sqrt{\sqrt[5]{x^4+\sqrt[5]{e^{4-x}}}}+\ln\|x-20,5\|$
3.	$e^{\sqrt{x+2}}\left(\frac{1}{7}+\ln\sqrt{x}\right)\frac{1}{3,5+x}$
4.	$(\sqrt{x}\sin x^2-1,3)\frac{1}{\sqrt[3]{x}+e^{2x}+\|\cos x\|}$
5.	$\sqrt{e^{\sin^2 x}}+2\ln 3x+\frac{1}{6}$
6.	$\left(\sqrt{1+x^2}+\frac{\ln^3 x}{1,6+x^4}\right)\sin 5x$
7.	$\sqrt{1/5+\sqrt[3]{e^x}}\frac{1}{\|\ln x^3+1,3\|}$
8.	$1,6+\ln\left\|4\frac{2}{7}\cdot\operatorname{tg}\sin\frac{5}{3}x\right\|$
9.	$1,1e^{-x}+\|\cos\sqrt{\pi x}\|-3/8$
10.	$\frac{1}{3}\sqrt{\|\sin x\|}\sqrt[3]{e^{0,12x}}$
11.	$2\pi x-\|\sin\sqrt{10,5x}\|\frac{1}{\sqrt[3]{x^2}+1/7}$
12.	$\ln\left(\sqrt{\|2-x\|}+1,2\right)\frac{1}{2+e^{-x}}+\sqrt[3]{2/x}$
13.	$\sqrt[5]{e^{-2+x}}\frac{1}{\sqrt{x^2+x^4+\ln\|x-3,14\|}}$
14.	$\left(\sqrt{\sin^3\frac{x}{2}}+\sqrt[3]{e^{1,3x}+e^{-1,3x}}\right)\frac{1}{\|x+5/2\|}$
15.	$(\|x\ln x-4\|\sqrt{x})\frac{1}{\sqrt[5]{e^{4x-1}}}$
16.	$\sqrt[3]{e^{2x}\sqrt{x}-\frac{x+1/3}{x}}\|\cos 2,5x\|$
17.	$\frac{x^3}{\sqrt{3}}-e^x\ln\|1,3^3+x^3\|+\frac{4}{3}$
18.	$\frac{\|7,2-10x\|}{\sqrt[3]{x^2+e^x}}\operatorname{arctg}\frac{4(x/3)}{\sqrt{1,1^3+x^2}}$
19.	$\frac{\ln\sqrt{\pi}+\|2-x\|}{3+1/x}+\sqrt[3]{x^2}\sin 1,4x$
20.	$(\sqrt[3]{\ln^2 x}+\operatorname{tg}\cos\pi x)\left\|\ln\frac{x}{10,5}+\frac{1}{3}\right\|$

Подготовить дисплейную карту и провести вычисление в дисплейном классе. Те же вычисления провести на персональной вычислительной технике (ВТ).

Задание 3. Написать программу ввода чисел с приглашением и вывода их с текстом. Сопроводить выдачу фамилией пользователя. Вывод организовать в столбец и в строку.

Номер варианта	Числа				
1.	$I=-37$	$A=0{,}3\cdot10^{-9}$	$EPS=0{,}001$	$K1=13$	$CR=0{,}27345\cdot10^{-5}$
2.	$J=5765$	$B=1{,}7\cdot10^{-15}$	$Q=0{,}273$	$L2=56$	$QW=1{,}37654\times10^{-10}$
3.	$K=24$	$C=3{,}4\cdot10^{-3}$	$DEL=14{,}745$	$M=123$	$AB=-2{,}97927\times10^{-5}$
4.	$L=-375$	$D=-1{,}2\cdot10^{8}$	$QOR=285{,}15$	$N=54$	$BC=3{,}84046\cdot10^{3}$
5.	$M=0$	$E=-2{,}37\cdot10^{5}$	$FI=-1{,}235$	$I=-748$	$CD=-4{,}73054\times10^{4}$
6.	$N=-1475$	$F=0{,}27\cdot10^{16}$	$PSI=3{,}412$	$J=0$	$DE=5{,}62033\cdot10^{5}$
7.	$IR=10$	$G=-0{,}15\cdot10^{-8}$	$TAU=0{,}01$	$K=-10$	$EF=-6{,}51027\cdot10^{6}$
8.	$JA=245$	$H=274{,}1\cdot10^{6}$	$AKOR=-27{,}13$	$L=265$	$FG=-7{,}46035\times10^{-7}$
9.	$KA=-31001$	$X=-0{,}1\cdot10^{-10}$	$SOL=14{,}11$	$M=100$	$GH=8{,}35048\cdot10^{-8}$
10.	$L1=15$	$Y=0{,}146\cdot10^{-5}$	$SW=1{,}234$	$N=-14$	$HO=9{,}24059\cdot10^{2}$
11.	$M1=-175$	$Z=14{,}1\cdot10^{3}$	$SQ=-99{,}11$	$I=697$	$OP=-0{,}13064\times10^{11}$
12.	$N1=-152$	$U=-1{,}23\cdot10^{-14}$	$DER=0{,}001$	$J=-18$	$PQ=-1{,}02037\times10^{-3}$
13.	$I=11$	$V=0{,}999\cdot10^{-9}$	$ARM=-0{,}02$	$K=1000$	$QR=2{,}91082\cdot10^{-5}$
14.	$J=101$	$W=-1{,}2\cdot10^{16}$	$SOR=9{,}743$	$L=1$	$RS=-3{,}00039\times10^{-6}$
15.	$K=-3145$	$O=1{,}0\cdot10^{-5}$	$TOR=1456{,}1$	$M=-20$	$ST=-4{,}75095\times10^{12}$
16.	$L=27$	$P=-2{,}5\cdot10^{13}$	$TER=27{,}27$	$N=3$	$TU=5{,}66046\cdot10^{-2}$
17.	$M=-9$	$R=75{,}1\cdot10^{-4}$	$AQ=0{,}034$	$I=175$	$UF=-6{,}49048\times10^{1}$
18.	$N=526$	$S=30{,}1\cdot10^{15}$	$BEQ=15{,}37$	$J=10000$	$FX=7{,}37099\cdot10^{-7}$
19.	$II=-10101$	$T=2{,}0\cdot10^{-6}$	$BEL=0{,}0995$	$K=-27$	$XY=-8{,}20010\times10^{8}$
20.	$JJ=2$	$X1=-4{,}0\cdot10^{-5}$	$REQ=-14{,}1$	$L=3421$	$YZ=9{,}10015\cdot10^{-9}$

Подготовить дисплейную карту и провести сеанс в дисплейном классе.

Задание 4. Написать программу:

1) табулирования значений функции $f(x)$ (задание 2) на интервале $[x_0 x_0+H]$, с шагом h

$$y_i=f(x_0+ih),\quad i=0,\ 1,\ \ldots,\ [H/h],$$

x_0, H, h задаются вводом, на терминал вывести таблицу с текстом $x=$, $y=$

Номер варианта	x_0	H	h	Номер варианта	x_0	H	h
1.	0,3	2	0,1	**11.**	0,3	0,7	0,02
2.	0,5	1,5	0,05	**12.**	0,4	0,8	0,02
3.	0,4	0,7	0,01	**13.**	0,2	0,5	0,01
4.	0,2	0,5	0,01	**14.**	0,6	0,9	0,01
5.	0,6	0,9	0,05	**15.**	0,2	0,6	0,02
6.	0,5	0,9	0,01	**16.**	0,3	0,6	0,025
7.	0,5	0,8	0,05	**17.**	0,4	0,7	0,025
8.	0,1	0,5	0,025	**18.**	0,5	0,8	0,02
9.	0,2	0,6	0,025	**19.**	0,5	0,9	0,01
10.	0,3	0,6	0,01	**20.**	0,2	0,6	0,02

2) Написать программу определения максимального и минимального значений y_i с выдачей на терминал.

3) Написать программу вычисления суммы S ряда с точностью ε, значение ε ввести. Вывести S и число слагаемых h, при котором достигнута точность ε;

$$S=\sum_{n=1}^{\infty}(-1)^{n+1}a_n.$$

Номер варианта	a_n	Номер варианта	a_n	Номер варианта	a_n	Номер варианта	a_n
1.	$\frac{1}{n}$	**6.**	$\frac{\cos(\pi/n)}{n}$	**11.**	$\frac{1}{n\ln(n+1)}$	**16.**	$\frac{1}{n^2}$
2.	$\frac{1}{n!}$	**7.**	$\frac{1}{\ln(n+1)}$	**12.**	$\sin\frac{\pi}{n}$	**17.**	$\left(\arcsin\frac{\pi}{n}\right)^2$
3.	$\frac{1}{2^n}$	**8.**	$\frac{1}{n(n+2)(n+3)}$	**13.**	$\operatorname{tg}\frac{\pi}{n}$	**18.**	$\left(\operatorname{tg}\frac{\pi}{n}\right)^4$
4.	$\frac{1}{n(n+1)}$	**9.**	$\frac{1}{\sqrt{n}}$	**14.**	$\arcsin\frac{\pi}{n}$	**19.**	$\frac{\ln n}{n}$
5.	$\frac{\sin(\pi/n)}{n^3}$	**10.**	$\frac{1}{\sqrt[3]{n^2}}$	**15.**	$\frac{1}{\operatorname{ch} n}$	**20.**	$\frac{e^{-n}}{n^3}$

Следует учесть, что в знакопеременном ряде точность ε определяется первым отброшенным членом.

Подготовить дисплейную карту и провести вычисления в дисплейном классе.

Задание 5. Написать программу вычисления матрицы X и свободного вектора b в системе уравнений $Xa=b$

$$X=(X_{i,j})=\sum_{k=0}^{N} x_k^{i+j}, \quad 0\leqslant i\leqslant m, \quad 0\leqslant j\leqslant m;$$

$$b=(b_i)=\sum_{k=0}^{\infty} x_k^i y_k, \quad 0\leqslant i\leqslant m,$$

по заданным массивам x_k, y_k, ($k=0, 1, 2, \ldots, N$); массивы вводить с терминала; матрицу X, вектор b выводить на терминал.

Номер варианта	Массивы						m
1.	x_k	0,1	0,3	0,4	0,6	0,7	0,9	3
	y_k	−1,0	0,5	0,8	1,7	2,5	2,1	
2.	x_k	−1	−0,5	0,1	0,4	0,8		2
	y_k	1,0	2,2	1,7	0,8	0,3		
3.	x_k	1,1	1,2	1,4	1,7	2,0	2,1	3
	y_k	−2,0	−1,8	−1,3	−1,0	−0,5	−0,6	
4.	x_k	−1	−0,5	0,0	0,3	0,7		2
	y_k	0,9	0,7	0,4	0,8	1,0		
5.	x_k	3	3,2	3,4	3,7	3,9	4,0	3
	y_k	−14	−10	−8	−12	−16	−18	
6.	x_k	1,0	3,0	7,0	10,0	14,0		2
	y_k	0,3	0,7	0,9	1,0	2,0		
7.	x_k	−10,0	−8,0	−5,0	−2,0	0,0	1,0	3
	y_k	6,0	3,0	0,0	−4,0	−2,0	0,0	
8.	x_k	2,0	3,0	5,0	6,0	8,0	9,0	3
	y_k	0,7	1,2	2,2	3,0	2,0	3,0	
9.	x_k	0,7	1,2	2,2	3,0	3,1		2
	y_k	0,8	1,0	1,3	1,2	1,4		
10.	x_k	100	110	125	130	140	150	3
	y_k	0,01	0,03	0,08	0,12	0,10	0,09	
11.	x_k	−10	−8	−5	−2	1		2
	y_k	3	4	0	−2	−1		
12.	x_k	1	4	9	15	17	20	3
	y_k	0,1	−0,2	−0,3	0,0	0,1	−0,2	
13.	x_k	2,0	3,1	4,2	5,6	6,4		2
	y_k	−15	−10	−8	−6	−7		
14.	x_k	−4	−3	−2	0	1	2	3
	y_k	3	4	5	4	3	1	
15.	x_k	10,5	11,5	12,5	13,0	14,0		2
	y_k	−6	−7	−5	0	2		
16.	x_k	2	4	6	8	10	12	3
	y_k	−3	−2	0	0	2	3	
17.	x_k	−0,3	−0,1	0,0	0,4	0,5		2
	y_k	5	5	3	5	5		
18.	x_k	−7,1	0,2	3,4	5,6	7,2	8,3	3
	y_k	−4	−2	−2	0	1	2	
19.	x_k	1	2	3	4	5		2
	y_k	10	20	15	20	10		
20.	x_k	−5	−4	−2	0	1	3	3
	y_k	0,2	0,25	0,23	0,19	0,16	0,12	

Подготовить дисплейную карту, провести вычисления в дисплейном классе и на персональной ВТ.

Задание 6. Написать функцию-подпрограмму, определяющую по матрице $X=(X_{i,j})$, $1<i, j<m$, вектору $b=b_i$, $i=1, 2, \ldots, m$,

Номер варианта	Q	Номер варианта	Q
1.	$\sum_{i,\,j=1}^{m} x_i^2$	11.	$\sqrt{\sum_{i=1}^{m} b_i^2}$
2.	$\sum_{i,\,j=1}^{m} \lvert x_{i,\,j}\rvert$	12.	$\max_{i,\,j} \lvert x_{i,\,j}\rvert$
3.	$\sum_{i=1}^{m} \lvert x_{i,\,i}\rvert$	13.	$\min b_i$
4.	$\max_{i} \sum_{j=1}^{m} x_{i,\,j} b_j$	14.	$\max_{j} \sum_{i=1}^{m} x_{i,\,j} b_i$
5.	$\min_{i} \sum_{j=1}^{m} x_{i,\,j} b_j$	15.	$\min_{i,\,k} \sum_{j=1}^{m} x_{i,\,j} x_{j,\,k}$
6.	$\max_{i,\,j} x_{i,\,j}$	16.	$\min_{i,\,j} x_{i,\,j}$
7.	$\sum_{i=1}^{m} \left(\sum_{j=1}^{m} x_{i,\,j} b_j \right)^2$	17.	$\max_{i} \left(\sum_{j=1}^{m} x_{i,\,j} + b_i \right)$
8.	$\max_{i,\,j} x_{i,\,j} + \max_{i} b_i$	18.	$\max_{i} \lvert b_i\rvert$
9.	$\min_{i,\,j} \lvert x_{i,\,j}\rvert$	19.	$\max_{i} \left(\max_{j} x_{i,\,j} + b_i\right)$
10.	$\max_{i} \min_{j} x_{i,\,j}$	20.	$\min_{i} \left(\max_{j} x_{i,\,j} + b_i\right)$

величину Q. В главной программе ввести с терминала X, b, обратиться к функции-подпрограмме, результат выдать на терминал. Значения для X, b взять из решения задачи задания 5.

Задание 7. Написать подпрограмму, определяющую по матрицам X, Y матрицу или вектор Z:

$$X_{i,\,j},\quad Y_{i,\,j},\quad Z_{i,\,j}(z_i),\quad 1<i,\ j<m.$$

В главной программе вычислить элементы $X_{i,\,j}$, $Y_{i,\,j}$ по формулам

$$x_{i,\,j}=\left(\sin\frac{\pi}{i}\right)+\left(\cos\frac{\pi}{j}\right),\quad 1\leqslant i,\ j\leqslant m,$$

$$y_{i,\,j}=(\cos\pi i)(\cos\pi j),$$

обратиться к подпрограмме, результат выдать на терминал.

Подготовить дисплейную карту и провести вычисления в дисплейном классе.

Задание 8. Написать программу, определяющую количество точек на плоскости, расположенных внутри фигуры, ввести с терминала координаты следующих 10 точек:
Результат вывести на терминал.

Таблица к заданию 7

Номер варианта	m	z	Номер варианта	m	z
1.	5	$z_{i,j}=\sum\limits_{k=1}^{m} x_{i,k}y_{k,j}$	**11.**	3	$z_i=\sum\limits_{j=1}^{m}\lvert x_{i,j}\rvert$
2.	7	$z_{i,j}=x_{i,j}+y_{i,j}$	**12.**	4	$z_j=\min\limits_{i}\lvert y_{i,j}\rvert$
3.	10	$z_{i,j}=\max\limits_{i,j}\lvert x_{i,j}\rvert$	**13.**		$z_{i,j}=\sum\limits_{k=1}^{m} y_{i,k}y_{k,j}$
4.	6	$z_i=\sum\limits_{j=1}^{m} y_{i,j}$	**14.**	6	$z_j=\max\limits_{i} x_{i,j}+\min\limits_{i} y_{i,j}$
5.	8	$z_{i,j}=\sum\limits_{k=1}^{m} x_{i,k}y_{k,j}+y_{i,j}$	**15.**	5	$z_i=x_{i,1}+y_{i,5}$
6.	4	$z_{i,j}=\sum\limits_{k=1}^{m} x_{i,k}x_{k,j}$	**16.**	3	$z_i=\max\limits_{j}\sum\limits_{k=1}^{m} x_{i,k}y_{k,j}$
7.	9	$z_j=\min\limits_{i} y_{i,j}$	**17.**	4	$z_i=\sum\limits_{k=1}^{m} x_{i,k}^2$
8.	6	$z_i=\max\limits_{j}(x_{i,j}+y_{i,j})$	**18.**	7	$z_i=\sum\limits_{k=1}^{m} y_{i,k}^2$
9.	5	$z_{i,j}=\sum\limits_{k=1}^{n} y_{i,k}x_{k,j}$	**19.**	6	$z_i=\sum\limits_{k=1}^{m} x_{i,k}^2+y_{i,k}^2$
10.	7	$z_i=\min\limits_{j} x_{i,j}+\max\limits_{j} y_{i,j}$	**20.**	4	$z_{i,j}=\lvert x_{i,j}\rvert+\lvert y_{i,j}\rvert$

Таблица к заданию 8

m	1	2	3	4	5	6	7	8	9	10
x	0	0,1	0,5	1,5	−0,8	1,5	−1,8	−0,5	0,9	1,1
y	0	0,2	−0,3	1,0	−0,6	1,2	0,3	0,5	−1,3	2,3

Номер варианта	Описание фигуры
1.	$(x^2+y^2\leqslant 1)$ и $(x\geqslant 0,\ y\leqslant 1-x,\ y\geqslant x-1)$
2.	$(\lvert x\rvert\leqslant 2,\ \lvert y\rvert\leqslant 1)$ и $(x^2+y^2\geqslant 1)$
3.	$(0\leqslant x\leqslant 2,\ 0\leqslant y\leqslant 2)$ или $(\lvert x\rvert\leqslant 1,\ \lvert y\rvert\leqslant 1)$
4.	$(x^2/4+y^2\leqslant 1)$ и $(\lvert x\rvert\geqslant 1)$
5.	$(\lvert x\rvert\leqslant 2,\ \lvert y\rvert\leqslant 1)$ или $(\lvert x\rvert\leqslant 1,\ \lvert y\rvert\leqslant 2)$
6.	$(-2\leqslant x\leqslant 0,\ 0\leqslant y\leqslant 2)$ или $(0\leqslant x\leqslant 2,\ \lvert y\rvert\leqslant 1)$
7.	$(\lvert x\rvert\leqslant 1,\ 0\leqslant y\leqslant 2)$ и $(x^2+y^2\geqslant 1)$

Номер варианта	Описание фигуры
8.	$(\|x\|\leqslant 2,\ \|y\|\leqslant 2)$ и $(\|x\|+\|y\|\geqslant 1)$
9.	$(x^2/4+y\leqslant 1)$ или $(x^2+y^2/4\leqslant 1)$
10.	$(x^2+y^2\leqslant 2)$ и $(x^2/4+y^2\leqslant 1)$
11.	$(x^2+y^2\leqslant 2)$ и $(x^2+y^2\geqslant 1)$
12.	$(\|x\|\leqslant 1,\ 0\leqslant y\leqslant 2)$ или $(\|x\|+\|y\|\leqslant 1)$
13.	$(x^2+y^2\leqslant 4)$ и $(\|x\|+\|y\|\geqslant 1)$
14.	$(x^2+y^2\leqslant 4,\ y\leqslant 0)$ и (внешность $\|x\|\leqslant 1,\ -1\leqslant y\leqslant 0$)
15.	$(x^2+y^2\leqslant 4)$ и (внешность $\|x\|\leqslant 1,\ \|y\|\leqslant 1$)
16.	$(x^2+y^2\leqslant 1)$ и $(x\geqslant 0,\ y\geqslant 0,\ y\leqslant 1-x)$
17.	$(x^2+y^2\leqslant 4)$ и (внешность $x^2+y^2\geqslant 1$) и $(\|x\|\leqslant 1)$
18.	$(x^2+y^2/4)$ и $(\|y\|\leqslant 1)$
19.	$(\|x\|+\|y\|\leqslant 2)$ и $(\|x\|\leqslant 1)$
20.	$(\|x\|+\|y\|\leqslant 2)$ и $(\|y\|\leqslant 1)$

Решить задачу графически, построив фигуру на плоскости. Подготовить дисплейную карту и провести вычисления в дисплейном классе.

Задание 9. Построить интерполяционный полином Лагранжа $L(x)$ по таблице задания 5. Вычислить приближенное значение $F(x)$ с помощью $L(x)$ в точке $x=\dfrac{x_2+x_3}{2}$, выполнить вычисления на персональной ВТ.

Оценить погрешность вычислений, если известно, что все производные $F(x)$ имеют оценку

$$|F^{(k)}(x)|\leqslant 1.$$

Принять, что абсолютная погрешность всех чисел таблицы задания 5 равна 0,001.

Задание 10. Приблизить функцию, заданную таблично (задание 5) полиномом $P_2(x)$ и $P_3(x)$ методом наименьших квадратов. Вычислить приближенное значение $F(x)$ с помощью $P_2(x)$ и $P_3(x)$ в точке $x=\dfrac{x_2+x_3}{2}$, выполнить вычисления на персональной ВТ. Написать программу, определяющую коэффициенты $P_3(x)$, $P_4(x)$ методом наименьших квадратов (вычисление нормальной системы Гаусса, обращение к библиотечной программе решения системы линейных уравнений) и вычисляющую с помощью $P_3(x)$ и $P_4(x)$ приближенное значение $F(x)$ в точке $x=\dfrac{x_2+x_3}{2}$.

На терминал выдать значения коэффициентов полинома, значения $F(x)$. Подготовить дисплейную карту и провести вычисления в дисплейном классе.

Задание 11. Проинтегрировать функцию $f(x)$ на заданном интервале $[a, b]$ с точностью ε. Точность оценивать по правилу Рунге.

Номер варианта	a	b	ε	$f(x)$	Используемый метод
1.	-1	0	10^{-2}	$x\cos x$	П, С
2.	0	1	$0{,}5\cdot 10^{-2}$	$x\sin x$	Т, С
3.	1	2	10^{-2}	$\sqrt{1+x}$	П, Т
4.	-1	0	10^{-2}	$1+x^2$	Т, С
5.	2	3	$0{,}5\cdot 10^{-2}$	$\ln x$	П, С
6.	0	1	10^{-2}	$\operatorname{tg} x$	Т, С
7.	-2	-1	10^{-2}	$\sin x\cos x$	П, Т
8.	3	4	$0{,}5\cdot 10^{-2}$	e^{-x}	Т, С
9.	$-0{,}5$	0,5	10^{-2}	x^2e^x	Т, С
10.	1	2	$0{,}5\cdot 10^{-2}$	$(\ln x)^2$	П, С
11.	4	5	10^{-2}	$\sin(\ln x)$	П, С
12.	0	1	10^{-2}	$\cos(\ln x)$	Т, С
13.	-1	0	10^{-2}	x^3e^{-x}	П, Т
14.	0	1	$0{,}5\cdot 10^{-2}$	$\sqrt{1+x^2}$	Т, С
15.	-1	0	10^{-2}	$\sin^2 x\cos x$	Т, С
16.	2	3	10^{-2}	$\sin^3 x$	П, С
17.	1	2	10^{-2}	$x^2\cos x$	П, С
18.	0	1	$0{,}5\cdot 10^{-2}$	$x^2\sin x$	Т, П
19.	-1	0	10^{-2}	$\operatorname{tg}^2 x$	Т, С
20.	3	4	10^{-2}	$\operatorname{ctg} x$	П, С

Вычислить точное значение интеграла по таблицам интегралов. Определить точность приближенных вычислений. Сравнить с правилом Рунге.

Написать программу вычисления интеграла по формуле трапеций, закончить вычисления по условию

$$|J_h - J_{h/2}| \leqslant 10^{-2}\varepsilon.$$

Структура программы — три модуля.

1) В главном: ввод a, b, ε; вывод $J_{h/2}$, если достигнута точность $10^{-2}\varepsilon$; обращение к подпрограмме вычислений формулы трапеций;

2) подпрограмма вычислений формулы трапеций;

3) функция-подпрограмма вычисления $f(x)$.

Подготовить дисплейную карту.

Задание 12. Провести вычисления в дисплейном классе интеграла по программе задания 11, а также с использованием библиотечной программы по методу Гаусса.

Задание 13. Решить с точностью $\varepsilon = 10^{-3}$ систему линейных уравнений методом простой итерации на персональной ВТ

$$\begin{cases} -0{,}1x_1 - 15x_2 + 0{,}1x_3 = b_1, \\ 10x_1 + 0{,}1x_2 - 0{,}05x_3 = b_2, \\ x_1 - 0{,}02x_2 + 20x_3 = b_3. \end{cases}$$

Оценить погрешность решения, если элементы матрицы и свободного вектора заданы с абсолютной погрешностью 0,01.

Номер варианта	b_1	b_2	b_3	Номер варианта	b_1	b_2	b_3
1.	5,0	7,0	17,0	**11.**	6,0	8,0	16,0
2.	5,1	7,1	16,9	**12.**	6,1	8,1	15,9
3.	5,2	7,2	16,8	**13.**	6,2	8,2	15,8
4.	5,3	7,3	16,7	**14.**	6,3	8,3	15,7
5.	5,4	7,4	16,6	**15.**	6,4	8,4	15,6
6.	5,5	7,5	16,5	**16.**	6,5	8,5	15,5
7.	5,6	7,6	16,4	**17.**	6,6	8,6	15,4
8.	5,7	7,7	16,3	**18.**	6,7	8,7	15,3
9.	5,8	7,8	16,2	**19.**	6,8	8,8	15,2
10.	5,9	7,9	16,1	**20.**	6,9	8,9	15,1

Задание 14. Написать подпрограмму решения систем линейных уравнений методом Зейделя. Подготовить дисплейную карту для решения системы уравнений задания 13 методом Зейделя и библиотечной программой решения систем линейных уравнений. Провести вычисления в дисплейном классе, оценить погрешность полученных решений.

Задание 15. Решить систему нелинейных уравнений методом простой итерации и методом Ньютона с точностью ε, выполнить вычисления на персональной ВТ

Номер варианта	Система уравнений	Точность ε
1.	$x_1=0{,}25\sin(0{,}3x_2)+2;\ x_2=0{,}5\cos 0{,}5x_1$	10^{-3}
2.	$x_1=0{,}5\sin(0{,}3x_2)+2;\ x_2=0{,}5\cos 0{,}3x_1$	10^{-4}
3.	$x_1=0{,}5\sin(0{,}2x_2)+2;\ x_2=0{,}5\cos 0{,}5x_1$	10^{-5}
4.	$x_1=0{,}5\sin(0{,}2x_2)+3;\ x_2=0{,}5\cos 0{,}5x_1$	10^{-5}
5.	$x_1=0{,}25\sin(0{,}3x_2)+2;\ x_2=\cos 0{,}3x_1$	10^{-3}
6.	$x_1=0{,}25\sin(0{,}3x_2)+3;\ x_2=0{,}1\cos 0{,}25x_1$	10^{-4}
7.	$x_1=0{,}5\sin(0{,}3x_2)+1;\ x_2=0{,}2\cos 0{,}25x_1$	10^{-4}
8.	$x_1=0{,}5\sin(0{,}2x_2)+3;\ x_2=0{,}3\cos 0{,}25x_1$	10^{-5}
9.	$x_1=0{,}25\sin(0{,}3x_2)+1;\ x_2=0{,}5\cos 0{,}3x_1$	10^{-3}
10.	$x_1=0{,}5\sin(0{,}3x_2)+2;\ x_2=0{,}5\cos 0{,}5x_2$	10^{-4}
11.	$x_1=0{,}5\sin(0{,}2x_2)+3;\ x_2=0{,}1\cos 0{,}2x_2$	10^{-4}
12.	$x_1=0{,}25\sin(0{,}25x_2)+1;\ x_2=0{,}2\cos 0{,}4x_2$	10^{-5}
13.	$x_1=0{,}3\sin(0{,}1x_2)+2;\ x_2=0{,}3\cos 0{,}4x_2$	10^{-3}
14.	$x_1=0{,}4\sin(0{,}2x_2)+3;\ x_2=0{,}1\cos 0{,}1x_2$	10^{-4}
15.	$x_1=0{,}3\sin(0{,}5x_2)+1;\ x_2=0{,}2\cos 0{,}3x_2$	10^{-3}
16.	$x_1=0{,}2\sin(0{,}25x_2)+2;\ x_2=0{,}3\cos 0{,}1x_2$	10^{-5}
17.	$x_1=0{,}4\sin(0{,}1x_2)+2;\ x_2=0{,}1\cos 0{,}2x_2$	10^{-4}
18.	$x_1=0{,}1\sin(0{,}2x_2)+3;\ x_2=0{,}1\cos 0{,}3x_2$	10^{-4}
19.	$x_1=0{,}2\sin(0{,}3x_2)+2;\ x_2=0{,}4\cos 0{,}3x_2$	10^{-3}
20.	$x_1=0{,}3\sin(0{,}4x_2)+1;\ x_2=0{,}3\cos 0{,}2x_2$	10^{-4}

Задание 16. Написать программу решения задачи задания 15 с помощью библиотечной подпрограммы. Провести вычисления в дисплейном классе для трех значений начальных векторов

$$\left(x_1^{(0)}=0,\ x_2^{(0)}=0\right);\ \left(x_1^{(0)}=2,\ x_2^{(0)}=-3\right);\ \left(x_1^{(0)}=10^2,\ x_2^{(0)}=10^2\right).$$

Предусмотреть выдачу на терминал последовательных приближений $(x_1^n,\ x_2^n)$, $n=0,\ 1,\ 2,\ \ldots$.

Задание 17. Минимизировать функцию $F(x_1,\ x_2)$ методом перебора в квадрате $(0\leqslant x_1\leqslant 1,\ 0\leqslant x_2\leqslant 1)$ с точностью определения точки минимума $\varepsilon=10^{-2}$. Ту же задачу решить методом градиентного спуска в квадрате $(|x_1|\leqslant 2,\ |x_2|\leqslant 2)$

$$F(x_1,\ x_2)=Ax_1+Bx_2+\exp\left(cx_1^2+Dx_2^2\right).$$

Номер варианта	A	B	C	D	Номер варианта	A	B	C	D
1.	1,2	−1,4	0,01	0,11	**11.**	11,2	−0,4	1,00	0,21
2.	2,2	−1,3	0,04	0,12	**12.**	12,2	−0,3	1,21	0,22
3.	3,2	−1,2	0,02	0,13	**13.**	13,2	−0,2	1,44	0,23
4.	4,2	−1,1	0,16	0,14	**14.**	14,2	−0,1	1,69	0,24
5.	5,2	−1,0	0,25	0,15	**15.**	15,2	0,0	1,96	0,25
6.	6,2	−0,9	0,36	0,16	**16.**	16,2	0,0	1,99	0,26
7.	7,2	−0,8	0,49	0,17	**17.**	17,2	0,1	2,56	0,27
8.	8,2	−0,7	0,64	0,18	**18.**	18,2	0,2	2,89	0,28
9.	9,2	−0,6	0,81	0,19	**19.**	19,2	0,3	3,24	0,29
10.	10,2	−0,5	0,94	0,20	**20.**	20,2	0,4	3,81	0,30

1) Написать программу перебора, реализующую два этапа: 1-й перебор — определение точки минимума с точностью $\varepsilon=10^{-1}$; 2-й перебор (в окрестности найденной точки минимума) — определение точки минимума с точностью $\varepsilon=10^{-2}$.

2) Написать программу минимизации градиентным спуском с помощью библиотечной подпрограммы. Подготовить дисплейную карту.

Задание 18. Провести вычисления в дисплейном классе по двум программам. В качестве начальных приближений для градиентного спуска взять четыре различные точки и точку, найденную перебором. Все точки спуска вывести на терминал.

Задание 19. Дано дифференциальное уравнение, описывающее линейный колебательный процесс

$$\ddot{x}+\omega^2 x=f(t)$$

с вынуждающей силой $f(t)$ на интервале $[a,\ b]$. В точке $t=a$ задаются начальные условия $x(a)=x_0$, $\dot{x}(a)=y_0$.

1) Записать аналитическое решение $x(t)$.

2) Разбить интервал $[a,\ b]$ узлами t_i с шагом $h=(b-a)/10$.

3) Используя формулу аналитического решения, вычислить точное решение в узлах на персональной ВТ. Построить график точного решения.

4) Найти приближенное решение в узлах x_i^h, $x_{2i}^{h/2}$ методом Эйлера. Найти погрешность в точке $t=b$ сравнением с точным значением и по правилу Рунге. Построить график приближенного решения.

5) Написать программу интегрирования для нелинейного колебательного процесса

$$\ddot{x}+\omega^2 x+\delta x^3=f(t),\ f(t)=\sin\omega_1 t.$$

Номер варианта	a	b	x_0	y_0	ω	ω_1	δ	ε
1.	-1	1	1,0	$-1,0$	2π	1	0,1	10^{-3}
2.	0	1	$-1,0$	0,0	$1,5\pi$	1	$-0,2$	10^{-4}
3.	1	2	0,5	$-0,5$	$1,8\pi$	2	0,2	10^{-4}
4.	-1	0	$-0,5$	1,0	$0,5\pi$	3	0,8	10^{-3}
5.	2	3	2,0	0,0	π	2	$-0,3$	10^{-4}
6.	0	1	0,0	$-1,0$	2π	1	0,6	10^{-3}
7.	2	-1	1,0	1,0	$1,5\pi$	3	1,0	10^{-4}
8.	3	4	$-2,0$	1,0	$1,8\pi$	1	$-1,0$	10^{-3}
9.	$-0,5$	0,5	0,1	0,3	$0,5\pi$	2	0,7	10^{-4}
10.	1	2	$-0,6$	1,0	π	1	0,4	10^{-4}
11.	4	5	2,0	3,0	2π	3	0,2	10^{-4}
12.	0	1	$-3,0$	0,0	$1,5\pi$	2	0,1	10^{-4}
13.	-1	0	1,0	2,0	$1,8\pi$	2	$-0,3$	10^{-3}
14.	0	1	2,0	3,0	$0,5\pi$	1	$-0,1$	10^{-3}
15.	2	3	$-0,2$	0,1	π	1	$-0,5$	10^{-4}
16.	1	2	0,8	1,2	2π	4	1,0	10^{-3}
17.	0	1	2,1	$-3,0$	$1,5\pi$	3	0,6	10^{-4}
18.	-1	0	0,5	0,5	$1,8\pi$	1	0,7	10^{-3}
19.	3	4	$-1,0$	$-1,0$	$0,5\pi$	1	$-0,8$	10^{-4}
20.	0	1	0,4	0,8	π	1	0,2	10^{-4}

Использовать библиотечную программу. Задать начальный шаг h, точность ε. Подготовить дисплейную карту.

Задание 20. Провести вычисления по программе задания 19 в дисплейном классе. Для выдачи графика нелинейного колебательного процесса на терминал (АЦПУ) использовать программу типа PRPLOT (*Отнес Р., Эноксон Л.* Прикладной анализ временных рядов.— М.: Мир, 1982. С. 387).

Задание 21. Дано дифференциальное уравнение, описывающее процесс управления

$$y''+p(x)y''+q(x)y=u(x),$$

где $u(x)$ — функция управления на интервале $[a, b]$. Требуется найти управляемый процесс $y(x)$ такой, что

$$y(a)=d_0,\ y(b)=d_1.$$

Здесь x играет роль времени.

1) Разбить интервал $[a, b]$ узлами x_i с шагом $h=(b-a)/5$, построить разностную схему для определения приближенного решения в узлах $x_i \to y_i$.

2) Методом прогонки решить полученную трехдиагональную систему линейных уравнений для h и $h/2$. Найти погрешность по правилу Рунге. Выполнить вычисления на персональной ВТ. Построить график для $y_i^{h/2}$.

Номер варианта	a	b	d_0	d_1	$p(x)$	$q(x)$	$u(x)$
1.	-1	0	1	1	$\sin x$	$-(1+x^2)$	e^x
2.	0	1	2	3	$\ln(1+x)$	$-e^{-x}$	1
3.	1	2	3	1	$\cos x$	-5	$x\sin x$
4.	-1	0	-1	2	2^x	$-(1+x^2)$	2
5.	2	3	2	3	x^2	$-x\ln x$	$x^2\cos x$
6.	0	1	3	4	$1+x$	-4	$\sin(2+x)$
7.	-2	-1	2	1	$1+e^x$	$-5-\sin^2 x$	3
8.	3	4	3	2	$\sin x$	-4	3
9.	$-0{,}5$	0,5	0	-2	x	-10	3^x
10.	1	2	1	1	$\cos^2 x$	$-7+\cos x$	$\ln(1+x^2)$
11.	4	5	2	1	$-e^x$	-6	$\sin x$
12.	0	1	1	4	$x\cos x$	$-e^x$	$1+x$
13.	-1	0	3	2	$\sin x$	$\cos x-5$	1
14.	0	1	-1	-1	e^x	-2	x
15.	2	3	2	3	x^3	-6	
16.	1	2	-3	2	$\ln(1+x)$	$\sin x-8$	1
17.	0	1	-2	1	$(1+x)^2$	-3	$2\cos 2x$
18.	-1	0	3	1	$\ln(1+x^2)$	$-(1+x^3)$	-3
19.	3	4	2	2	x^2	$-10+x$	e^{-x}
20.	0	1	3	3	$\sin x$	-4	$\cos x$

3) Написать программу решения краевой задачи с использованием библиотечной подпрограммы. Подготовить дисплейную карту.

Задание 22. Провести вычисления по программе задания 21 в дисплейном классе. Оценить погрешность вычислений по правилу Рунге. Построить график приближенного решения.

Задание 23. Найти стационарное распределение температуры в прямоугольной пластине, все стороны которой поддерживаются при нулевой температуре. Пластина нагревается постоянным током, выделяющим в единице объема тепло Q. Задача сводится к решению уравнения Пуассона в прямоугольнике $(0 \leqslant x \leqslant a,\ 0 \leqslant y \leqslant b)$

$$\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = -\frac{Q}{k}$$

(где k — коэффициент внутренней теплопроводности) с условиями Дирихле

$$u(0,\ x)=0,\quad u(b,\ x)=0,\quad u(y,\ 0)=0,\quad u(y,\ a)=0.$$

Здесь $u(x,\ y)$ — температура пластины в точке x, y. Аналитическое решение задачи (см. гл. 5)

$$u(x,\ y)=\frac{Q}{2k}\left[x(a-x)-\frac{8a^2}{\pi^3}\sum_{n=0}^{\infty}\frac{\sin\dfrac{(2n+1)\pi x}{a}}{(2n+1)^3}\cdot\frac{\operatorname{ch}\dfrac{(2n+1)\pi(b-2y)}{2a}}{\operatorname{ch}\dfrac{(2n+1)\pi b}{2a}}\right].$$

1) Разбить область G сеткой с шагом $h_x={}^{a}/_{3}$, $h_y={}^{b}/_{3}$. Построить разностную схему. Найти приближенное решение в узлах. Вычисления выполнить на персональной ВТ. Построить график в оксонометрической проекции.

2) Вычислить в одном из узлов значение точного решения, используя аналитическую формулу.

Номер варианта	a	b	Q/k	Номер варианта	a	b	Q/k
1.	0,5	0,5	1,0	**11.**	0,4	0,5	2,0
2.	0,8	1,0	1,1	**12.**	0,8	0,1	1,9
3.	1,0	0,2	1,2	**13.**	0,4	0,3	1,8
4.	1,0	0,8	1,3	**14.**	0,6	1,0	1,7
5.	0,2	0,3	1,4	**15.**	0,2	1,0	1,6
6.	0,8	0,1	1,5	**16.**	0,8	1,0	1,5
7.	0,2	1,0	1,6	**17.**	0,3	0,2	1,4
8.	0,3	0,4	1,7	**18.**	1,0	0,8	1,3
9.	0,5	0,5	1,8	**19.**	1,0	0,2	1,2
10.	1,0	0,2	1,9	**20.**	0,2	0,5	1,1

3) Написать программу решения задачи с помощью библиотечной подпрограммы. Положить, что изменение мощности источников тепла от координаты пластины описывается законом

$$k^{-1}Q(x, y)=a(x^2+y^2)+b.$$

Абсолютная погрешность вычислений $\varepsilon=10^{-2}$.

Задание 24. Провести вычисления по программе задания 23 в дисплейном классе. Построить график приближенного решения.

Задание 25. Найти изменение распределения температуры со временем в тонком однородном стержне. На концах стержня задаются температурные режимы. Начальная температура всего стержня постоянная.

Задача сводится к решению уравнения теплопроводности

$$\frac{\partial u}{\partial t}=\varkappa^2\frac{\partial^2 u}{\partial x^2},$$

где $\varkappa^2$ — коэффициент температуропроводности, $0\leqslant t\leqslant t_1$, $0\leqslant x\leqslant a$;

$$u(x, 0)=u_0,\ u(0, t)=\mu_0(t),\ u(a, t)=\mu_1(t),$$

$0\leqslant t\leqslant t_1$, где $u(x, t)$ — температура стержня в точке x в момент времени t;

$$\mu_0(0)=\mu_1(0)=u_0.$$

Аналитическое решение задачи (см. гл. 5) имеет вид

$$u(x, t)=\mu_0(t)+\frac{x}{a}(\mu_1(t)-\mu_0(t))+$$

$$+\frac{2}{a}\int_0^t\int_0^a\sum_{n=1}^{\infty}e^{-\left(\frac{\pi n}{a}\right)^2\varkappa^2(t-\tau)}\sin\frac{\pi n}{a}x\sin\frac{\pi n}{a}\xi\left(\frac{\xi}{a}(\dot\mu_0-\dot\mu_1)-\dot\mu_1\right)d\xi\,d\tau.$$

где $\dot\mu$ означает дифференцирование по τ.

1) Построить двухслойную явную схему, приняв $h_x=a/10$; h_t выбрать из условия устойчивости. Вычислить два временных слоя на персональной ВТ. Построить графики.

2) Вычислить точное значение температуры в пятом узле на втором временном слое, используя аналитическую формулу. Взять в сумме 3—4 слагаемых, результат сравнить.

Номер варианта	a	t_1	$\varkappa^2$	u_0	$\mu_0(t)$	$\mu_1(t)$
1.	1,2	2,0	1,1	10	$u_0+0{,}1\sin 2\pi t$	$u_0+0{,}2\sin 4\pi t$
2.	1,1	2,1	1,2	11	$u_0+0{,}2t$	$u_0-0{,}3te^{-t}$
3.	0,8	2,2	1,3	12	$u_0+0{,}1te^{-t}$	$u_0-0{,}1\sin 2\pi t$
4.	0,6	2,3	1,4	13	$u_0+0{,}4t$	$u_0+0{,}3\sin \pi t$
5.	1,4	2,4	1,5	14	u_0-t	$u_0+\sin 2\pi t$
6.	1,6	2,5	1,6	15	$u_0+\sin 4\pi t$	u_0-2t
7.	0,9	2,6	1,7	16	u_0	$u_0+\sin \pi t$
8.	0,7	2,7	1,8	17	u_0e^{t}	u_0e^{-t}
9.	1,3	2,8	1,9	18	$u_0+\sin 3\pi t$	u_0
10.	1,2	2,9	2,0	19	$u_0+\sin \pi t$	u_0e^{-2t}
11.	1,1	3,0	1,9	20	u_0+2t	u_0-3t
12.	1,2	2,9	1,8	21	u_0e^{2t}	$u_0+\sin \pi t$
13.	1,3	2,8	1,7	22	u_0	u_0-4t
14.	1,4	2,7	1,6	23	u_0+te^{-t}	u_0
15.	1,5	2,6	1,5	24	u_0-3t	u_0e^{t}
16.	1,6	2,5	1,4	25	$u_0+4\sin 4\pi t$	$u_0-6\sin \pi t$
17.	1,5	2,4	1,3	24	u_0	u_0e^{-t}
18.	1,4	2,3	1,2	23	$u_0+3\sin \pi t$	u_0e^{2t}
19.	1,3	2,2	1,0	22	u_0e^{-2t}	u_0e^{-3t}
20.	1,2	2,1	0,9	21	$u_0+5\sin 2\pi t$	u_0

3) Написать программу решения задачи с помощью библиотечной подпрограммы. Выдачу на терминал организовать через пять временных слоев. Абсолютная погрешность вычислений $\varepsilon=10^{-4}$.

4) Провести вычисления в дисплейном классе. Построить графики временных слоев.

Задание 26. Найти изменение температуры $u(x, y, t)$ во времени для однородной квадратной пластины; на сторонах поддерживаются заданные температурные режимы. Начальное распределение температуры задано. Задача сводится к решению уравнения теплопроводности

$$\frac{\partial u}{\partial t}=\varkappa^2\left(\frac{\partial^2 u}{\partial x^2}-\frac{\partial^2 u}{\partial y^2}\right)$$

в области $\{0\leqslant x\leqslant 1,\ 0\leqslant y\leqslant 1,\ 0\leqslant t\leqslant t_1\}$. Начальное распределение для u: $u(x, y, 0)=x+y$. Краевые условия

$$u(x, y, t)\big|_{\substack{y=0\\0\leqslant x\leqslant 1}}=xe^{-at},\quad u(x, y, t)\big|_{\substack{y=1\\0\leqslant x\leqslant 1}}=(x+1)e^{-at},$$

$$u(x, y, t)\big|_{\substack{x=0\\0\leqslant y\leqslant 1}}=ye^{-at},\quad u(x, y, t)\big|_{\substack{x=1\\0\leqslant y\leqslant 1}}=(y+1)e^{-at}.$$

1) Построить явную и неявную разностные схемы (метод дробных шагов). Принять h_x, $h_y=0{,}1$; $0{,}01$; в явной схеме h_t выбрать из условия устойчивости и заданной точности. Значения параметров $\varkappa^2$, t_1, a взять из таблицы задания 25.

2) Составить блок-схему алгоритма для явной и неявной схем.

3) Написать программу решения задачи по явной схеме с точностью $\varepsilon = 10^{-3}$.

4) Провести вычисления в дисплейном классе. На терминал выдавать каждый 5-й временной слой. Построить график распределения температуры при $t = t_1$ в оксонометрической проекции.

Задание 27. Постановка технической задачи типового расчета принадлежит доценту МЭИ Ю. Ю. Зуеву.

Для перемещения захвата автоматического манипулятора батискафа применяется следящая система, упрощенная схема которой показана на рис. Т.1. Система состоит из следующих основных частей: рукоятки управления (РУ), блока формирования управляющего входного сигнала (БФУС), электронного усилителя (ЭУ), электромеханического преобразователя (ЭМП), золотникового гидроусилителя (ЗГУ), гидроцилиндра (ГЦ), шток которого соединен с захватом автоматического манипулятора (ЗАМ), а также цепи обратной связи (ОС), включающей датчик обратной связи (ДОС), блок усиления сигнала обратной связи (БУОС) и сумматор (СУМ).

Система работает следующим образом. При отсутствии управляющего входного сигнала $g(t)$ равно нулю, перемещение входной тяги БФУС $z(t)$, входное напряжение сумматора $u_{\text{вх}}(t)$, входное напряжение ЭУ $u_1(t)$, ток ЭМП $I(t)$, смещение золотника $x(t)$. Золотник занимает среднее положение относительно втулки ЗГУ. К средней расточке втулки ЗГУ от насоса подводится жидкость с высоким давлением $P_{\text{н}}$. Левая и правая проточки втулки соединяются с трубопроводом низкого давления $P_{\text{сл}}$, по которому жидкость поступает в гидравлический бак. При $x(t) = 0$ давления в обеих полостях гидродвигателя равны друг другу и их разность $P(t) = 0$. Поршень неподвижен и находится в положении, которое считается исходным: $y(t) = 0$. Напряжение $u_{\text{дос}}(t)$, снимаемое с датчика обратной связи, пропорциональное перемещению штока $y(t)$ с коэффициентом пропорциональности $k_{\text{дос}}$, а также напряжение БУОС $u_{\text{ос}}(t)$, являющееся усиленным $u_{\text{дос}}(t)$ с коэффициентом усиления $k_{\text{ос}}$, равны нулю. Система неподвижна.

При появлении входного управляющего сигнала, т. е. перемещении РУ, входная тяга БФУС перемещается на величину $z(t)$, пропорциональную $g(t)$, с коэффициентом пропорциональности k_1. Выходное напряжение БФУС $u_{\text{вх}}(t)$ вследствие значительного быстродействия усилителя допустимо считать пропорциональным $k_2 z(t)$. Рост тока управления ЭУ при увеличении $u_1(t)$ подчиняется соотношению

$$T_{\text{у}} \dot{I} + I = k_{\text{у}} u_1(t),$$

где $T_{\text{у}}$, $k_{\text{у}}$ — соответственно электромагнитная постоянная времени и коэффициент усиления ЭУ по току.

Наличие тока управления приводит к появлению электродвижущей силы $k_{\text{эмп}} I$, действующей на золотник. Эта сила, преодолевая инерционную составляющую силы сопротивления $T_2^2 \ddot{x}$,

составляющую силы вязкого трения $T_1\dot{x}$ и силу, пропорциональную смещению золотника, обусловливает перемещение последнего на величину $x(t)$ (например, вправо, как показано на рис. Т. 1). Под действием давления $P_н$ жидкость через щель, образованную кромками проточки во втулке и буртов золотника, поступает в полость гидроцилиндра (левую на рис. Т. 1). В результате давления в этой полости, действующего на площадь поршня F, последний перемещается (вправо на рис. Т. 1) с координатой $y(t)$, скоростью $\dot{y}$ и ускорением $\ddot{y}(t)$, выдавливая жидкость из другой полости гидроцилиндра (правой на рис. Т. 1) в ЗГУ и далее через щель (правую на рис. Т. 1) в трубопровод низкого давления $P_{сл}$. Количество жидкости, поступающее из ЗГУ в гидроцилиндр за единицу времени и равное величине $xk_3\sqrt{P_н - P_{сл} - P}$, частично расходуется на движение поршня со скоростью $\dot{g}(t)$. Часть расхода, определяемая как $T_3\dot{P}$, тратится на заполнение объемов в трубопроводах и гидроцилиндре, которые образуются в результате изменения давлений жидкости при ее прохождении по гидравлическому тракту системы. И наконец, часть расхода из ЗГУ, пропорциональная $\dot{P}$ с коэффициентом пропорциональности k_4, уходит через уплотнения, сальники, зазоры в трубопровод низкого давления.

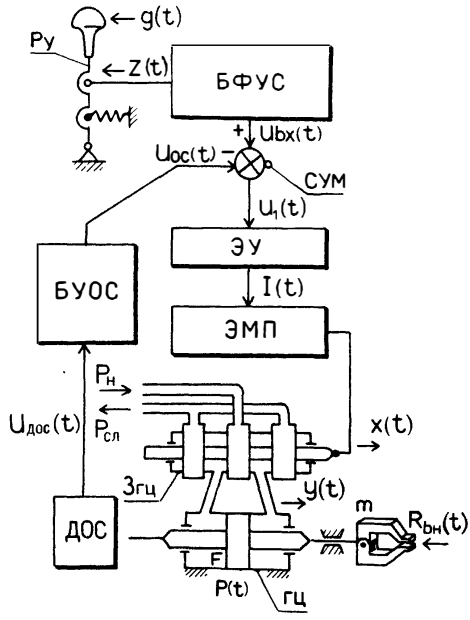

Рис. Т.1

При перемещении поршня и соединенного с ним ЗАМ преодолеваются инерционная составляющая силы сопротивления, пропорциональная массе подвижных частей m, и другие составляющие внешней нагрузки, объединенные в функционал, имеющий в данном случае вид

$$R_{вн} = k_5 y + k_6 \dot{y} + k_7 y^2 + k_8 \dot{y}^2.$$

Одновременно ДОС, входное звено которого соединено со штоком гидроцилиндра, вырабатывает напряжение датчика $u_{дос}$, которое затем усиливается БУОС до значения $u_{ос}$, поступающей на сумматор системы, где происходит ее вычитание из $u_{вх}(t)$. Напряжение $u_{ос}(t)$ пропорционально $u_{дос}(t)$, а последнее — смещению $y(t)$, поэтому разность напряжений $u_{дос}(t)$ обращается в нуль при условии $u_{вх}(t) = u_{ос}(t)$, т. е. при перемещении захвата на некоторую величину, пропорциональную $g(t)$ и $z(t)$. Уравнения, описывающие динамический процесс системы, имеют следующий вид:

уравнение РУ $z(t)=k_1 g(t)$;
уравнение БФУС $u_{вх}(t)=k_2 z(t)$;
уравнение СУМ $u_1(t)=u_{вх}(t)-u_{ос}(t)$;

уравнение ЭУ $T_у \frac{dI}{dt}+I=k_у u_1(t)$;

уравнение ЭМП $T_2^2 \frac{d^2x}{dt^2}+T_1 \frac{dx}{dt}+x=k_{эмп} I$;

уравнение ЗГУ $k_3 x\sqrt{P_н-P_{сл}-P}=F\frac{dy}{dt}+T_3\frac{dP}{dt}+k_4 P$;

уравнение гидроцилиндра $PF=m\frac{d^2y}{dt^2}+R_{вн}\left(t,\ y,\ \frac{dy}{dt}\right)$;

уравнение $R_{вн}$ $R_{вн}=k_5 y+k_6\frac{dy}{dt}+k_7 y^2+k_8\left(\frac{dy}{dt}\right)^2$;

уравнение ДОС $u_{дос}=k_{дос} y$;
уравнение БУОС $u_{ос}=k_{ос} u_{дос}$.

Ориентировочные значения параметров таковы:

$P_н=(10\div 20)\cdot 10^6$ Н/м^2; $k_1=0{,}1\div 0{,}5$; $k_5=5\cdot 10^3\div 5\cdot 10^4$ Н/м;

$F=(5\div 20)\cdot 10^{-4}$ м^2; $k_2=1\div 10$ В/м; $k_6=0\div 10^3$ Нс/м;

$m=50-500$ кг; $T_2=10^3-10^{-2}$ с; $k_7=0-10^5$ Н/м^2;

$k_у=0{,}01-0{,}05$ а/в; $T_1=0{,}2\cdot 10^{-3}-10^{-3}$ с; $k_8=0-10^5$ Нс^2/м^2;

$T_3=10^{-5}-10^{-4}$ с; $k_{эмп}=10^{-2}-5\cdot 10^{-2}$ м/а; $k_{дос}=1-10$ В/м;

$k_3=10^{-4}-10^{-6}$ м^{3}Н$^{-1/2}$/с; $k_4=0-10^{-12}$ м^5/Нс; $k_{ос}=1-10$;

$g(t)=g_0=0-0{,}5$ м.

Величины входных сигналов $u_{вх}(t)=0{,}01-0{,}1$ в.

Порядок установившихся значений переменных определяется по математической модели для $t\to\infty$.

Характерный вид получаемых сходящихся переходных процессов управления изображен на рис. Т. 2.

Критерии динамического качества системы (рис. Т. 2):

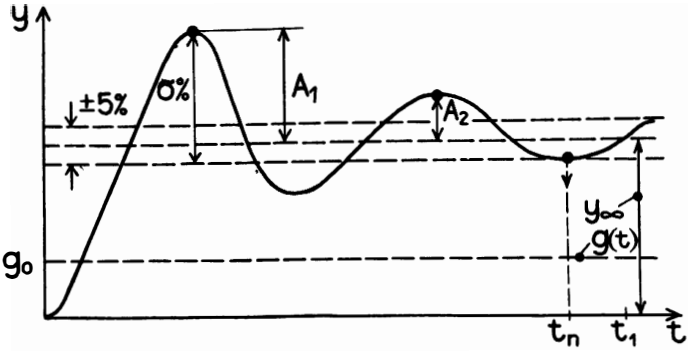

Рис. Т.2

Номер варианта	Критерий	Параметры	Номер варианта	Критерий	Параметры
1.	σ	F, $P_н$	**11.**	n	$P_н$, $k_{эмп}$
2.	$t_п$	F, k_8	**12.**	ξ	$P_н$, $k_у$
3.	n	F, $k_{дос}$	**13.**	σ	$P_н$, k_2
4.	ξ	F, k_1	**14.**	$t_п$	k_8, $k_{дос}$
5.	σ	F, $k_{эмп}$	**15.**	n	k_8, k_1
6.	$t_п$	F, $k_у$	**16.**	ξ	k_8, $k_{эмп}$
7.	n	F, k_2	**17.**	σ	k_8, $k_у$
8.	ξ	$P_н$, k_8	**18.**	$t_п$	k_8, k_2
9.	σ	$P_н$, $k_{дос}$	**19.**	n	$k_{дос}$, k_1
10.	$t_п$	$P_н$, k_1	**20.**	ξ	$k_{дос}$, $k_{эмп}$

σ%-перерегулирование, $\sigma = \left| \dfrac{y_{max} - y_\infty}{y_\infty} \right| 100$;

$t_п$ — время переходного процесса (время, за которое процесс полностью «входит» в ± 5%-ную трубку около y_∞);

n — число переходов процесса через асимптоту y_∞ до «входа» в 5%-ную трубку;

$\xi = \ln \dfrac{A_2}{A_1}$ — декремент затухания.

Техническая задача: исследовать качество переходного процесса в зависимости от параметров системы управления. Провести оптимизацию системы. При выполнении расчетов следует, варьируя заданные параметры, добиться сходящегося процесса и минимизировать какой-либо из заданных частных критериев качества: σ, $t_п$, n или ξ.

1) Определить входные и выходные данные задачи: константы, переменные. Указать их возможный диапазон изменения.

2) Привести уравнения, описывающие динамический процесс системы управления (переходный процесс), к каноническому виду

$$\frac{dy_i}{dt} = f_i(y_1, y_2, ..., y_n, t),$$

$$y_i(0) = y_i, \quad 1 \leqslant i \leqslant n, \quad 0 \leqslant t \leqslant t_1, \quad t_1 \text{ — задать.}$$

3) Привести каноническую систему дифференциальных уравнений к безразмерному виду, выполнить масштабирование.

4) Интегрируя систему дифференциальных уравнений и варьируя параметры системы уравнений, найти значения параметров, при которых имеет место сходящийся переходный процесс.

Расходящийся переходный процесс определяет решения, для которых существует t_*:

$$|y(t_*)| > Y, \quad 0 < t_* \leqslant t_1, \quad Y \text{ — задать.}$$

5) Построить графики сходящихся переходных процессов $y(t)$, $0 \leqslant t \leqslant t_1$, в зависимости от параметров системы управления (от двух параметров).

6) Найти значение критерия качества переходного процесса для нескольких значений параметров системы управления.

7) Вычислить оптимальное значение критерия качества, варьируя двумя параметрами системы управления.

8) Построить график оптимального переходного процесса.

9) Точность вычислений — относительная, равна 3%.

К отчету по типовому расчету следует представить:

1. Анализ технической задачи.
2. Постановку математической задачи: приведение ее к канонической форме.
3. Обоснование выбора метода интегрирования системы обыкновенных дифференциальных уравнений, выбора начального шага интегрирования.
4. Обоснование выбора метода оптимизации.
5. Блок-схему алгоритмов.
6. Фортран-программы.
7. Распечатки с выдачей результатов.
8. Графики переходных процессов.

Замечание. 1) Исследование переходного процесса (построение графика) рекомендуется проводить по безразмерной величине $\frac{y(t)}{y_\infty}$, где $y_\infty = \lim\limits_{t \to \infty} y(t)$ — установившееся положение захвата. Для нахождения y_∞ нужно решить систему нелинейных уравнений, получающуюся из системы дифференциальных уравнений заменой всех производных нулем.

2) Грубый выбор параметров сходящихся процессов из заданных диапазонов осуществляется решением линеаризованной системы уравнений переходного процесса.

Задание 28. Найти установившийся режим, решая нелинейную систему уравнений. Решить линеаризованную систему уравнений переходного процесса, определить грубо параметры сходящихся переходных процессов. Выбрать численный метод интегрирования дифференциальных уравнений с учетом значения коэффициента жесткости. Представить полную блок-схему алгоритма, используя приемы структурной алгоритмизации.

Задания 29—30. Написать программу в соответствии с блок-схемой задания 28, придерживаясь технологии структурного программирования. Провести отладку программы на ЭВМ.

Задания 31—32. Провести вычисления по программе заданий 29—30 на ЭВМ и подготовить отчет.

II УРОВЕНЬ

Задачи и упражнения к главе 1

II.1.1. Описать архитектуру используемой вычислительной машины.

II.1.2. Каковы технические характеристики центрального процессора, оперативной памяти и других уровней памяти используемой ЭВМ?

II.1.3. Какие устройства ввода-вывода применяются на ЭВМ?

II.1.4. Описать систему адресации используемой ЭВМ.

II.1.5. Какие элементы новых архитектурных решений присутствуют в Вашей ЭВМ?

II.1.6. Какова стоимость аренды 1 ч процессорного времени и дисплейного времени на Вашей ЭВМ?

II.1.7. Каким образом можно осуществить перенос результатов вычислений, данных, программ с Вашей ЭВМ на другую. Что для этого необходимо выполнить?

II.1.8. Может ли работать Ваш терминал в автономном режиме? Каковы его возможности в этом режиме?

Задачи и упражнения к главе 2

Спроектировать алгоритм указанного ниже численного метода, взяв за основу элементарный алгоритм гл. 3 так, чтобы он обладал некоторыми универсальными свойствами. Для этого следует включить в структуру алгоритма по крайней мере два блока: блок диагностики возможных ошибок (деление на ноль, зацикливание и т. п.) и блок контроля точности вычислений.

II.2.1. Метод градиентного спуска.

II.2.2. Метод Симптона численного интегрирования.

II.2.3. Метод прогонки.

II.2.4. Метод Рунге—Кутта 4-го порядка.

II.2.5. Метод прогонки решения краевой задачи для линейного дифференциального уравнения 2-го порядка.

II.2.6. Фурье-анализ.

II.2.7. Метод Ньютона решения нелинейных уравнений.

Указание. В качестве ориентира для создания проекта универсального алгоритма можно взять тексты фортран-программ соответствующих методов, входящие в стандартные библиотеки или из [15, 26, 27, 32].

Задачи и упражнения к главе 3

II.3.1. Решить задачи $1 \div 57$, представленные в [6, с. 25—44].

II.3.2. Написать программу, придерживаясь правил структурного программирования, реализующую проект алгоритма II.2.1—II.2.7. Комментарии в программе должны полностью описывать алгоритм и программу.

II.3.3. Написать программу с тестовой задачей для II.3.2.

Задачи и упражнения к главе 4

II.4.1. Описать все этапы прохождения фортран-программы в операционной системе Вашей ЭВМ. Подготовить краткую инструкцию для работы на машине.

II.4.2. Представить структуру файловой системы. Какова спецификация файлов системы. Указать соглашения, принимаемые по умолчанию.

II.4.3. Какие команды следует применить, чтобы переписать информацию с магнитной ленты на магнитный диск (кассетный, гибкий), магнитную ленту.

II.4.4. Представить основные команды текстового редактора. Подготовить краткую инструкцию для работы на машине.

II.4.5. Описать последовательность Ваших действий, если диагностика транслятора с фортрана указывает на допущенные ошибки.

II.4.6. Описать последовательность Ваших действий, если обнаружены ошибки во время выполнения программы.

II.4.7. Какие действия следует предпринять, если есть подозрение, что Ваша программа зациклилась.

II.4.8. Какие средства отладки фортран-программ имеются в Вашем распоряжении.

Задачи и упражнения к главе 5

II.5.1. Найти координаты, в которых у функции $y=x/(x^2+\varepsilon^2)^{1/2}$, $|\varepsilon| \ll 1$ в окрестности $x=0$ устраняется особенность по ε производной $\dfrac{dy}{dx}(x/\varepsilon)$.

II.5.2. Произвести масштабирование координат так, чтобы сильно вытянутый по оси z эллипсоид

$$x^2+y^2+\varepsilon^2 z^2=1, \quad \varepsilon \ll 1,$$

преобразовался в сферу. Каков смысл этого преобразования в задачах оптимизации?

II.5.3. Каков смысл перечисленных ниже чисел в математических моделях технических задач? Выбрать из них то число, которое встречается в Ваших моделях, дать его определение. Число: Маха, Рэлея, Пекле, Нуссельта, Струхала, Кнудсена.

II.5.4. Привести пример точно решаемой задачи и ее аналитическое решение из следующих семейств:

1) корни полиномов (степени $n \geqslant 5$);
2) суммирование рядов;
3) решение СЛУ (порядка $n \geqslant 10$);
4) решение ОДУ (краевая задача).

II.5.5. Показать, что:

$$\int_0^1 \frac{\sin \varepsilon t}{t} dt = \varepsilon - \frac{1}{18}\varepsilon^3 + O(\varepsilon^5), \quad \varepsilon \to 0;$$

$$\int_0^x t^{-3/4} e^{-t} dt = 4x^{1/4} - \frac{4}{5}x^{5/4} + O(x^{9/4}), \quad x \to 0.$$

II.5.6. На основе аналитического решения задачи II.5.4 построить два первых приближения к решению методом возмущений в соответствующей возмущенной задаче.

II.5.7. Как можно использовать аналитические решения предыдущих задач при построении численных методов в той области параметра возмущения, где методы возмущений неприменимы? Привести пример.

II.5.8. Перечислить методы оценки погрешности суммирования сходящегося ряда, выполняемого на ЭВМ. Как найти вклад погрешности вычислений в общую погрешность?

II.5.9. Привести пример ускорения серийных вычислений за счет предвычислений.

II.5.10. Привести пример ускорения серийных вычислений за счет перехода к целочисленной арифметике. Оценить вычислительную погрешность, связанную с таким переходом.

Задачи и упражнения к главе 6

Задана функция

$$y(x)=\sum_{n=1}^{100}\exp(\cos(n^{-1}x)),\quad 0\leqslant x\leqslant 1.$$

II.6.1. Построить интерполяционный параболический сплайн с погрешностью аппроксимации в равномерной норме $\varepsilon=10^{-3}$, $\varepsilon=10^{-4}$. Оценить ускорение вычисления значения $y(x)$ с помощью сплайна в сравнении с исходной формулой.

II.6.2. Решить задачу II.6.1 построением кубического сплайна.

II.6.3. Построить для $y(x)$ полином 1-й степени наилучшего равномерного приближения. Найти оценку погрешности в равномерной норме.

II.6.4. Построить для $y(x)$ полином наилучшего среднеквадратичного приближения с погрешностью аппроксимации $\varepsilon=10^{-3}$, $\varepsilon=10^{-4}$.

II.6.5. Чем определяется выбор нормы в задачах аппроксимации функций?

II.6.6. Используя библиотечную подпрограмму аппроксимации, написать программу вычисления значения функции $y(x)$ в произвольной точке x с заданной точностью ε.

Задачи и упражнения к главе 7

II.7.1. Привести пример функций $f(x)$, $g(x)$, имеющих в узлах x_i квадратурной формулы совпадающие значения $f(x_i)=g(x_i)$, точные интегралы которых отличаются на сколь угодно большую величину.

II.7.2. Определить наименьшее из возможных значений шага h интегрирования методом Симпсона на Вашей ЭВМ в арифметике одинарной точности.

II.7.3. Написать программу вычисления несобственных интегралов вида $\int_0^{\infty} f(x)dx$, в основу которой следует положить библиотечную подпрограмму вычисления $\int_0^{A} f(x)dx$ и алгоритм увеличения предела A с остановом по заданной точности. Провести вычисления для интегралов

$$\int_0^{\infty} e^{-x^2}dx, \quad \int_0^{\infty} \frac{\sin x}{1+x^2}dx.$$

II.7.4. Как оценить погрешность численного интегрирования в задаче II.7.3?

II.7.5. Определить, какая из библиотечных подпрограмм позволяет максимально далеко продвинуться по ω, $\omega \to \infty$, при вычислении интеграла

$$J_0(\omega) = \frac{1}{\pi}\int_0^{\pi} \cos(\omega \sin\theta)d\theta \sim \sqrt{\frac{2}{\pi\omega}}\cos\left(\omega - \frac{\pi}{4}\right)$$

с заданной относительной точностью 1%.

Задачи и упражнения к главе 8

Задана матрица

$$A = \begin{pmatrix} 3 & 1 & 2 \\ 4 & 3 & 2 \\ 1 & 4 & 2 \end{pmatrix}.$$

II.8.1. Показать, что число обусловленности матрицы не изменится, если ее умножить на ненулевую константу.

II.8.2. Определить число обусловленности матрицы A.

II.8.3. Найти LU-разложение матрицы A.

II.8.4. Определить погрешность решения системы линейных уравнений $Ax = b$ с вектором $b = (1, 2, 3)$, компоненты которого заданы с абсолютной погрешностью 0,05: 1) матрица A задана точно; 2) матрица A имеет абсолютную погрешность элементов 0,01. Погрешность вычислений не учитывать.

II.8.5. Решить СЛУ $Ax = b$ задачи II.8.4 методом простой итерации, погрешность в евклидовой норме $\varepsilon = 10^{-2}$.

II.8.6. Решить СЛУ $Ax = b$ задачи II.8.4 методом Зейделя, погрешность в евклидовой норме $\varepsilon = 10^{-2}$.

II.8.7. Найти собственные значения и собственные векторы матрицы A, применяя библиотечные подпрограммы. Записать решение ОДУ

$$\frac{dx}{dt} = Ax - b, \quad x(0) = 0.$$

II.8.8. Найти, используя библиотечную подпрограмму, минимум целевой функции

$$5x_1+2x_2-3x_3+x_4$$

при ограничениях

$$\begin{aligned} &2x_1-x_2+x_3+x_4\leqslant 5,\\ &x_1+x_2-x_3-x_4\leqslant 2,\\ &5x_1-8x_2+2x_3+4x_4\leqslant 3. \end{aligned}$$

Задачи и упражнения к главе 9

II.9.1. В основе активационного анализа состава вещества лежит поиск показателей экспонент a_i, $1\leqslant i\leqslant m$, и коэффициентов c_i, минимизирующих невязку

$$\|f(x)-\sum_{i=1}^{m} c_i\exp(a_i x)\|,$$

где $f(x)$ — экспериментально измеренная функция. Вывести систему уравнений, определяющих искомые c_i, a_i, если $\|\ \|$ — дискретная среднеквадратичная норма на узлах x_j, $1\leqslant j\leqslant n$.

II.9.2. Сколько вещественных, комплексных решений имеет система уравнений

$$x_1^2+x_2^2-4=0,\quad x_1^2-x_2+0{,}1\sin x_2=0?$$

II.9.3. Найти вещественные решения системы уравнений задачи II.9.2. методом простой итерации с абсолютной точностью $\varepsilon=10^{-3}$. Уточнить решения (точность $\varepsilon=10^{-6}$) методом Ньютона.

II.9.4. Найти комплексные решения системы уравнений задачи II.9.2.

II.9.5. Привести задачу II.9.2 к задаче минимизации целевой функции и решить ее, применяя библиотечную подпрограмму минимизации.

II.9.6. Показать, что метод градиентного спуска минимизации $\Phi(x_1, \ldots, x_n)$ эквивалентен методу Эйлера для системы ОДУ

$$\frac{dx}{dt}=-\operatorname{grad}\Phi(x),\quad x(0)=x^0.$$

II.9.7. Показать, что целевые функции $\Phi(x)$, поверхности уровней которых имеют вид сильно вытянутых эллипсоидов, приводят к жестким системам ОДУ (см. гл. 10) в задаче II.9.6.

Задачи и упражнения к главе 10

Задачи II.10.1—II.10.5 входят в типичный пакет тестов программ интегрирования задачи Коши для ОДУ. Требуется проинтегрировать уравнение или систему с заданной относительной локальной точностью $\delta=10^{-2}$, 10^{-4} на интервале $0\leqslant x\leqslant 20$ двумя библиотечными подпрограммами, дополнив их просчетом числа Q вычисле-

ний правых частей уравнений на всем интервале интегрирования. Сравнить эффективность подпрограмм по числу Q.

II.10.1. Одно уравнение

1) $y' = -y$, $y(0) = 1$;
2) $y' = -y^{3/2}$, $y(0) = 1$;
3) $y' = y\cos x$, $y(0) = 1$;
4) $y' = (y-x)/(y+x)$, $y(0) = 4$;
5) $y' = (y/4)(1-y/20)$, $y(0) = 1$.

II.10.2. Система малой размерности

1) $\begin{cases} y_1' = 2(y_1 - y_1 y_2), & y_1^{(0)} = 1, \\ y_2' = -(y_2 - y_1 y_2), & y_2^{(0)} = 3 \end{cases}$

(рост двух враждующих популяций);

2) $\begin{cases} y_1' = -y_1 + y_2, & y_1(0) = 2, \\ y_2' = y_1 - 2y_2 + y_3, & y_2(0) = 0, \\ y_3' = y_2 - y_3, & y_3(0) = 1 \end{cases}$

(линейная химическая реакция);

3) $\begin{cases} y_1' = -y_1, & y_1(0) = 1, \\ y_2' = y_1 - y_2^2, & y_2(0) = 0, \\ y_3' = y_2^2, & y_3(0) = 0 \end{cases}$

(нелинейная химическая реакция);

4) $y_1' = -y_2 + y_1 y_3/(y_1^2 + y_2^2)^{1/2}$, $y_1(0) = 3$,
$y_2' = y_1 - y_2 y_3/(y_1^2 + y_2^2)^{1/2}$, $y_2(0) = 0$,
$y_3' = y_1/(y_1^2 + y_2^2)^{1/2}$, $y_3(0) = 0$
(интегральная поверхность тора);

5) $y_1' = y_2 y_3$, $y_1(0) = 0$,
$y_2' = -y_1 y_3$, $y_2(0) = 1$,
$y_3' = -0{,}51 y_1 y_2$, $y_3(0) = 1$.
(эйлеровы уравнения движения твердого тела).

II.10.3. Система умеренной размерности

$$\begin{matrix} y_1' = \\ y_2' = \\ \vdots \\ y_{51}' = \end{matrix} \begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ 0 & & -2 & 1 \\ & & 1 & -2 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{51} \end{pmatrix}; \quad y(0) = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

II.10.4. Уравнение орбиты

$y_1' = y_3,\ y_1(0) = 1 - e,$
$y_2' = y_4,\ y_2(0) = 0,$
$y_3' = -y_1/(y_1^2 + y_2^2)^{3/2},\ y_3(0) = 0,$
$y_4 = -y_2/(y_1^2 + y_2^2)^{3/2},\ y_4(0) = (1+e)^{1/2}/(1-e)^{1/2},$
$e = 0{,}1$ — эксцентриситет орбиты.

II.10.5. Уравнение второго порядка, приведенное к системе

$$y_1' = y_2,\ y_1(0) = 2,$$
$$y_2' = (1 - y_1^2)y_2 - y_1,\ y_2(0) = 0$$

(уравнение Ван дер Поля, описывающее процессы в электронных схемах).

II.10.6. Для систем уравнений задач II.10.2—II.10.3 поставить краевые задачи, которые решить численно, используя библиотечные подпрограммы, с точностью $\delta = 10^{-2}$.

Задачи и упражнения к главе 11

II.11.1. Найти форму равновесия однородной прямоугольной мембраны, закрепленной по краям, если к ней приложено нормальное давление $P(x, y)$ на единицу площади. Указать те функции $P(x, y)$, для которых задача решается аналитически, численно. Решить задачу численно с относительной точностью 1%. Профиль мембраны представить в графической форме.

II.11.2. Задача об управлении тепловым процессом в стержне состоит в определении функции $g(t)$ такой, чтобы решение уравнения теплопроводности $u(t, x)$ — распределение температуры

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

с условиями

$$u(x, 0) = \sin x + \cos x,\ \ 0 \leqslant x \leqslant 1,$$

$$\frac{\partial u}{\partial x}(0, t) = e^{-t},\ \ \frac{\partial u}{\partial x}(1, t) = g(t),\ \ 0 \leqslant t \leqslant 1,$$

в момент времени $t = 1$ совпадало с заданным распределением температуры

$$u(1, x) = e^{-1}(\sin x + \cos x).$$

1) Найти точное решение задачи.

2) Найти численно решение, используя явную схему, с абсолютной точностью 10^{-3} в равномерной норме.

II.11.3. Задача об определении критической температуры детонирующего стержня описывается уравнением

$$\frac{\partial u}{\partial t}=\frac{\partial^2 u}{\partial x^2}+c_1 \exp\left(c_2-\frac{1}{u}\right)$$

в области $0\leqslant x\leqslant 1$, $0\leqslant t\leqslant T$; c_1, c_2 — константы с условиями $u(x, 0)=u_1$; $u(0, t)=u_0$, $u(1, t)=u_1$, u_0, u_1 — константы $(u_0>u_1)$. Критическая температура u_k определяется так, что при $u_0<u_k$ решение при $t\to\infty$ стремится к стационарному $u(\infty, x)\leqslant u_0$ для всех точек $0\leqslant x\leqslant 1$, в то время как при $u_0>u_k$ в некоторой точке $x_*\in(0, 1)$ имеет место неравенство $u(t, x_*)>u_0$ (происходит зажигание). Пусть $c_1=c_2=1$, $u_1=0{,}1$.

1) Найти величину u_0, приводящую к зажиганию, считая $t_\infty=T$. Использовать метод прямых с разностной аппроксимацией $\frac{\partial^2 u}{\partial x^2}$. Точность вычислений — относительная и составляет 1%. Интегрировать систему ОДУ библиотечной программой для жестких уравнений.

2) Определение u_0 выполнить перебором и методом деления отрезка $[0, u_1]$ пополам.

III УРОВЕНЬ

Задачи и упражнения к главе 1

III.1.1. Предложить архитектуру ЭВМ, наиболее подходящую для решения задач Вашей предметной области. Обосновать выбор и указать близкую по параметрам ЭВМ (из имеющихся на мировом рынке).

III.1.2. Указать недостатки грубой классификации архитектур, представленной четырьмя типами ЭВМ из **1.3**.

III.1.3. Какие средства графического отображения информации имеются на Вашей ЭВМ. Описать их возможности.

III.1.4. Каким образом можно расширить вычислительные ресурсы Вашей ЭВМ (подключение специальных процессоров (каких?); дополнительной памяти (какой?)). Обосновать экономическую целесообразность предлагаемого расширения.

Задачи и упражнения к главе 2

Спроектировать алгоритм указанного ниже численного метода, взяв за основу алгоритм (текст) соответствующего метода, входящего в стандартную библиотеку, так, чтобы он удовлетворял требованиям серийных вычислений. Для этого необходимо: 1) заменить по возможности наиболее емкие вычислительные блоки

блоками предвычислений (см. гл. 5); 2) описать структуру алгоритма; 3) описать поток данных алгоритма; 4) описать структуру программы.

III.2.1. Метод градиентного спуска.

III.2.2. Метод прогонки.

III.2.3. Метод решения жестких ОДУ.

III.2.4. Метод типа Ньютона решения нелинейных уравнений.

Задачи и упражнения к главе 3

III.3.1. Написать программу в соответствии с проектом алгоритма III.2.1—III.2.4, придерживаясь правил структурного программирования на языке фортран 77 в двух вариантах: 1) в одинарной точности; 2) в двойной точности.

III.3.2. Написать программы с тестовой задачей для III.3.1.

III.3.3. Провести вычисления на ЭВМ для теста и сравнить эффективность Вашей программы с библиотечной, сравнить объем необходимой памяти.

Задачи и упражнения к главе 4

III.4.1. Объем Вашей программы и/или данных превышает выделяемый пользователю ресурс памяти. Какой режим работы позволяет в Вашей ОС выполнять некоторые программы такого сорта? Привести пример программы, требующей больших ресурсов, и показать, как готовится ее выполнение.

III.4.2. Какие средства Вашей ОС позволяют составить цепочку команд (трансляция, компоновка, загрузка, исполнение и т. п.), а затем запустить всю цепочку одной командой?

III.4.3. Собрать из стандартных библиотек или из литературы [3, 9, 15, 26, 27, 30, 32], тексты фортран-программ, наиболее часто применяемых в Вашей работе по специальности в форме, удобной для ввода в ЭВМ. Подготовить личную библиотеку объектных файлов отобранных фортран программ.

Задачи и упражнения к главе 5

III.5.1. Какие невозмущенные решения используются в Ваших моделях, когда следующие числа велики (малы): число Маха, Рэлея, Пекле, Нуссельта, Струхала, Кнудсена и т. д. Привести пример.

III.5.2. Привести пример точно решаемой задачи и ее аналитическое решение из следующих семейств:

1) нелинейные системы уравнений;

2) собственные значения и собственные векторы матриц;

3) решение уравнений с частными производными.

III.5.3. Показать, что

$$J_0(x)=\frac{1}{\pi}\int_0^{\pi}\cos(x\sin\theta)\,d\theta\sim\sqrt{\frac{2}{\pi x}}\cos\left(x-\frac{\pi}{4}\right),\quad x\to\infty,$$

$$\int_0^1\ln x\,e^{i\lambda x}\,dx\sim-\frac{i\ln\lambda}{\lambda}-\frac{i\gamma+\pi/2}{\lambda},\quad \lambda\to\infty,$$

где $\gamma=-\int_0^{\infty}e^{-x}\ln x\,dx$.

III.5.4. На основе аналитического решения III.5.1 построить первое приближение к решению методом возмущения.

III.5.5. Арифметика с подсчетом числа значащих цифр состоит в указании после арифметической операции верных значащих цифр результата по числу верных значащих цифр операндов. Как можно реализовать программно такую арифметику на ЭВМ? В каких вычислениях она может быть полезна?

Задачи и упражнения к главе 6

III.6.1. Написать программу одномерной параболической сплайн-интерполяции, предназначенную для серийных вычислений. За основу взять программу из [9].

III.6.2. Написать программу одномерной кубической сплайн-интерполяции для серийных вычислений. За основу взять программу из [9].

III.6.3. На основе библиотечной подпрограммы двумерной аппроксимации написать программу графического вывода поверхности по заданным узловым точкам. Это элемент проектирования в диалоговом режиме поверхностей сложных форм [3].

Задачи и упражнения к главе 7

III.7.1. Вычислить интегралы следующих функций с интегрируемой особенностью, применяя аналитические и численные методы и используя библиотечные подпрограммы.

$$\int_0^1\ln x\,e^x\,dx,\quad\int_0^1\frac{e^{-x^3}}{\sqrt{x}}\,dx.$$

Абсолютная точность вычислений $\varepsilon=10^{-4}$, $\varepsilon=10^{-5}$.

III.7.2. Правило Рунге применяется с фиксированным шагом h. Привести пример функции, для которой оценка погрешности по правилу Рунге неприменима для $h\sim10^{-2}$ и применима для $h\sim10^{-6}$.

III.7.3. Написать программу вычисления интеграла с выбором локального шага по заданной точности в соответствии с алгоритмом, изложенным в 7.3. Показать (на тестах) примеры функций, для которых программа оказывается эффективнее библиотечных подпрограмм интегрирования (каких?).

III.7.4. Используя библиотечную подпрограмму интегрирования функций двух переменных, написать программу вычисления координат центра масс, моментов инерции неоднородных плоских фигур.

Задачи и упражнения к главе 8

III.8.1. Привести пример СЛУ, для которой метод простой итерации сходится, а метод Зейделя расходится.

III.8.2. Привести пример СЛУ, для которой метод Зейделя сходится, а метод простой итерации расходится.

III.8.3. Написать программу метода исключения Гаусса с выбором ведущего элемента по строкам на языке фортран 77 в арифметике двойной точности.

III.8.4. Написать программу задачи III.8.3 в арифметике целых чисел.

III.8.5. Написать программу определения областей локализации собственных значений матрицы A с выводом границ областей на графический дисплей или графопостроитель.

Задачи и упражнения к главе 9

III.9.1. Определить, какой сложности целевые функции $\Phi(x)$ (размерность x, объем арифметических операций) и с какой точностью определения точки минимума можно оптимизировать на Вашей ЭВМ перебором.

III.9.2. Написать программу минимизации целевой функции, объединяющую алгоритм грубого перебора с дальнейшим уточнением точки минимума наискорейшим градиентным спуском.

III.9.3. Написать программу поиска $\min\Phi(x)$ интегрированием системы ОДУ

$$\frac{dx}{dt} = -\operatorname{grad}\Phi(x), \quad x_0 = x^0$$

с контролем коэффициента жесткости (см. гл. 10) и выбором соответствующего метода интегрирования.

Задачи и упражнения к главе 10

Задачи III.10.1—III.10.5 входят в типичный пакет тестов программ интегрирования задачи Коши для жестких ОДУ. Требуется проинтегрировать уравнения с заданной локальной относительной

точностью $\delta=10^{-2}$, $\delta=10^{-4}$ на интервале $0\leqslant x\leqslant x_1$ с начальным шагом h_0 двумя библиотечными подпрограммами, одна из которых ориентирована на жесткие уравнения. Определить число Q вычислений правых частей уравнений на интервале $0\leqslant x\leqslant x_1$. Сравнить эффективность подпрограмм по числу Q.

III.10.1. Линейные системы с вещественными собственными значениями:

1) $y_1'=-0{,}5y_1$, $y_1(0)=1$, $x_1=20$, $h_0=10^{-2}$,
$y_2'=-y_2$, $y_2(0)=1$,
$y_3'=-100y_3$, $y_3(0)=1$,
$y_4'=-90y_4$, $y_4(0)=1$;
2) $y_1'=-1800y_1+900y_2$, $y_1(0)=0$, $x_1=120$, $h_0=5\cdot10^4$,
$y_i'=y_{i-1}-2y_i+y_{i+1}$, $y_i(0)=0$, $2\leqslant i\leqslant 8$,
$y_9'=1000y_8-2000y_9+1000$, $y_9(0)=0$.

III.10.2. Линейные системы с комплексными собственными значениями:

1) $y_1'=-y_1+y_2$, $y_1(0)=1$, $x_1=20$, $h_0=7\cdot10^{-3}$,
$y_2'=-100y_1-y_2$, $y_2(0)=0$,
$y_3'=-100y_3+y_4$, $y_3(0)=1$,
$y_4'=-100\,000y_3-100y_4$, $y_4(0)=0$;
2) $y_1'=-10y+\alpha y_2$, $y_1(0)=1$, $x_1=20$, $h_0=10^{-2}$,
$y_2'=-\alpha y_1-10y_2$, $y_2(0)=1$, $\alpha=3;\ 8;\ 25;\ 100$,
$y_3'=-4y_3$, $y_3(0)=1$,
$y_4'=-y_4$, $y_4(0)=1$,
$y_5'=-0{,}5y_5$, $y_5(0)=1$,
$y_6'=-0{,}1y_6$, $y_6(0)=1$.

III.10.3. Нелинейные системы, эквивалентные линейным:

1) $y_1'=-y_1+y_2^2+y_3^2+y_4^2$, $y_1(0)=1$; $x_1=20$; $h_0=10^{-2}$,
$y_2'=-10y_2+10(y_3^2+y_4^2)$, $y_2(0)=1$,
$y_3'=-40y_3+40y_4^2$, $y_3(0)=1$,
$y_4'=-100y_4+2$, $y_4(0)=1$;
2) $y_1'=-y_1+2$, $y_1(0)=1$; $x_1=20$; $h_0=10^{-2}$,
$y_2'=-10y_2+\beta y_1^2$, $y_2(0)=1$; $\beta=0{,}1;\ 1;\ 10;\ 20$,
$y_3'=-40y_3+4\beta(y_1^2+y_2^2)$, $y_3(0)=1$,
$y_4'=-100y_4+10\beta(y_1^2+y_2^2+y_3^2)$, $y_4(0)=1$.

III.10.4. Нелинейные системы с вещественными собственными значениями якобиана:

1) $y_1'=0{,}2(y_2-y_1)$, $y_1(0)=0$; $x_1=400$; $h_0=1{,}7\cdot10^{-2}$,
$y_2'=10y_1-(60-0{,}125y_3)y_2+0{,}125y_3$, $y_2(0)=0$,
$y_3'=1$, $y_3(0)=0$
(моделирование атомного реактора);

2) $y_1' = -0{,}04y_1 + 0{,}01y_2y_3$, $y_1(0) = 1$; $x_1 = 40$; $h_0 = 10^{-5}$,
$y_2' = 400y_1 - 100y_2y_3 - 3000y_2^2$, $y_2(0) = 0$,
$y_3' = 30y_2^2$, $y_3(0) = 0$
(химическая модель)

3) $y_1' = -0{,}013y_1 - 1000y_1y_3$, $y_1(0) = 1$; $x_1 = 50$; $h_0 = 2{,}9 \cdot 10^{-4}$,
$y_2' = -2500y_2y_3$, $y_2(0) = 1$,
$y_3' = -0{,}013y_1 - 1000y_1y_3 - 2500y_2y_3$, $y_3(0) = 1$
(химическая модель)

III.10.5. Нелинейные системы с комплексными собственными значениями якобиана:

1) $y_1' = y_2$, $y_1(0) = 2$, $x_1 = 1$, $h_0 = 10^{-3}$,
$y_2' = 5(1 - y_1^2)y_2 - y_1$, $y_2(0) = 0$
(уравнение Ван дер Поля)

2) $y_1' = -(55 + y_3)y_1 + 65y_2$, $y_1(0) = 1$, $x_1 = 500$, $h_0 = 0{,}02$,
$y_2' = 0{,}0785(y_1 - y_2)$, $y_2(0) = 1$,
$y_3' = 0{,}1y_1$, $y_3(0) = 0$
(физическая модель)

III.10.6. Для систем уравнений III.10.1—III.10.5 поставить краевые задачи, которые решить численно с точностью $\delta = 10^{-2}$.

Задачи и упражнения к главе 11

III.11.1. Подстановка

$$u(t, x) = -2a^2 \ln v(t, x)$$

сводит нелинейное уравнение

$$\frac{\partial u}{\partial t} + \frac{1}{2}\left(\frac{\partial u}{\partial x}\right)^2 = a^2 \frac{\partial^2 u}{\partial x^2}$$

к линейному

$$\frac{\partial v}{\partial t} = a^2 \frac{\partial^2 v}{\partial x^2}.$$

Указать, какие задачи для нелинейного уравнения можно решить аналитически, численно и каким образом, используя указанную подстановку.

III.11.2. Тримолекулярная модель химической кинетики (брюсселятор) описывается системой уравнений

$$\frac{\partial u}{\partial t} = a - (b+1)u + u^2v + d_1 \frac{\partial^2 u}{\partial x^2},$$

$$\frac{\partial v}{\partial t} = bu - u^2v + d_2 \frac{\partial^2 v}{\partial x^2},$$

где a, b, d_1, d_2 — константы. Решение $u(t, x)$, $v(t, x)$ ищется в области $(0 \leqslant t \leqslant T,\ 0 \leqslant x \leqslant 1)$. Граничные условия $u(t, 0) = u(t, 1) = a$; $v(t, 0) = v(t, 1) = b/a$.

1) Линеаризовав уравнения заменой

$$u=a+u_1,\ v=b/a+v_1,$$

найти для линейных уравнений относительно u_1, v_1 условия, при которых

$$u_1(t, x)\to 0,\ v_1(t, x)\to 0,\ t\to\infty,$$

для любых начальных условий $u_1(0, x)$, $v_1(0, x)$.

2) В области параметров (a, b) потери устойчивости численным интегрированием найти предельные $(t\to\infty)$ решения $(u(\infty, x)$, $v(\infty, x))$ для различных начальных условий $u(0, x)$, $v(0, x)$.

Значения констант в расчетах принять $a\cong 2$, $d_1=1{,}6\cdot 10^{-3}$, $d_2=6\cdot 10^{-3}$.

АНГЛО-РУССКИЙ СЛОВАРЬ

ACCEPT
принять, ввести

ACCESS
доступ

ACCESS MODE
режим доступа

ADDITION
сложение

ADDRESS
адрес

ALLOCATION
распределение памяти

.AND.
логическая операция .И.

APPLICATION PROGRAM
прикладная программа

ARGUMENT
аргумент

ARRAY
массив

ARRAY REFERENCE
ссылка на массив

ASSIGN
назначить

BACKSPACE
возврат к началу предыдущей записи

BATCH PROCESSING
пакетная обработка

BIT
бит

BLOCKDATA
блок данных

BLOCK SIZE
размер блока

BUFFER
буфер

BYTE
байт

CALL
вызвать

CHARACTER
символ

CLOSE
закрыть

COMMAND
команда

COMMON
общий

COMPILER
компилятор

COMPLEX
комплексный

CONTINUATION LINE
строка продолжения

CONTINUE
продолжить

CONTROL CHARACTER
управляющий символ

COUNTER
счетчик

CPU
центральный процессор

CURSOR
курсор

DATA
данные

DEBUGGING
отладка

DEFINE FILE
определение размера и структуры файла

DELETE
удалить

DIAGNOSTICS
диагностика

DIMENSION
размер

DIRECT ACCESS
прямой доступ

DIRECTORY
каталог

DISK
диск

DISPLAY
терминал с электронно-лучевой трубкой

DIVISION
деление

DO
делать

DOUBLE PRECISION
двойная точность

EDIT
редактирование

EDITOR
редактор

ELSE
иначе

ELSEIF... THEN
иначе, если ... то

END
конец

ENDFILE
признак конца файла

ENDIF
конец если

END OF BLOCK (EOB)
конец блока

END OF FILE (EOF)
конец файла

ENTRY
вход

.EQ.
операция отношения .равно.

EQUIVALENCE
эквивалентность

.EQV.
логическая операция .равно.

ERASE CHARACTER
символ удаления

ERROR
ошибка

ERROR MESSAGE
сообщение об ошибке

EXECUTE
выполнение

EXIT
выход

EXPONENT
порядок

EXTERNAL
внешний

FALSE
ложь

FILE
запись

FIND
найти

FLOATING-POINT OPERATION
операции с плавающей точкой

FORMAT
формат

FORMAT SPECIFICATION
спецификация формата

FUNCTION
функция

.GE.
операция отношения .больше или равно.

GO TO
идти к

.GT.
операция отношения .больше чем.

HARDWARE
аппаратное обеспечение ЭВМ

IF
если

IF ... THEN
если ... то

IMPLICIT
неявный

I/O
ввод — вывод

INPUT
ввод

INQUIRE
опросить

INSTRUCTION
команда

INTEGER
целый

INTERRUPT
прерывание

INTRINSIC
внутренний

JUMP
переход к другой команде

LABEL
метка

.LE.
операция отношения .меньше или равно.

LIBRARY
библиотека

LIST
список

LISTING
распечатка

LOCATION
ячейка

LOGICAL
логический

LOOP
цикл

LOOP NESTING
вложенные циклы

LOSS
потеря

.LT.
операция отношения .меньше чем.

MAIN
главный

MAP
карта, схема

MARK
маркер, знак

MEGAFLOP
мегафлоп $=10^6$ оп/с с плавающей точкой

MEMORY
память

MESSAGE
сообщение

MULTIPLICATION
умножение

NAME
имя

.NE.
операция отношения .не равно.

.NEQV.
логическая операция .не равно.

.NOT.
логическая операция отрицание .не.

NUMBER
число

OBJECT CODE
объектный код

OPEN
открыть

.OR.
логическая операция .или.

OUTPUT
вывод

OVERFLOW
переполнение

PAGE
страница

PARAMETER
параметр

PARANTHESES
скобки

PASSWORD
пароль

PAUSE
пауза

PERMISSION
разрешение

PRECISION
точность

PRINT
печатать

PRINTER
печатающее устройство

PRIORITY
приоритет

PROGRAM
программа

READ
читать

REAL
вещественный

REAL-TIME
реальное время

RECORD
запись

RECORD FORMAT
формат записи

RETURN
возврат

REWIND
возврат к началу файла

ROOT
корень

ROUNDING
округление

ROUTINE
стандартная, системная программа

RUN
прогон, запуск (программы)

SAVE
сохранять

SEQUENCE
последовательность

SIGN
знак, символ, признак

SOFTWARE
программное обеспечение ЭВМ

STATEMENT
оператор

STOP
стоп

STORAGE
память

STRING
строка

SUBROUTINE
подпрограмма

SUBTRACTION
вычитание

SYMBOL
символ, знак

TAPE
лента

TIME
время

TIME SHARING
разделение времени

TRANSLATION
перевод

TRAP
прерывание

TRUE
истина

TYPE
напечатать

UNDERFLOW
потеря точности

UNIT
устройство

USER
пользователь

VALUE
значение, величина

VARIABLE
переменная величина

VOLUME
том

WARNING
предупреждение

WORD
слово

WRITE
писать

ZERO
нуль

Фортран 77

Текст программ на фортране состоит из строк.

Операторы занимают позиции 7—72 в строке.

Символы: буквы верхнего регистра A—Z, цифры 0—9, спец. символы = + − * / ( ) ′ . $, : пробел.

Символы C или * в первой позиции обозначают строку — комментарий.

Символ, отличный от нуля и пробела в 6-й позиции, означает, что строка есть продолжение предыдущей.

Метка оператора помещается в позиции 1—5.

Применяемые обозначения

⟨...⟩ = ⟨элемент, включаемый в программу пользователем⟩
[...] = [необязательная часть оператора]
{...} = {последовательность из нуля или более элементов}
⟨список⟩ = ⟨элемент списка, элемент списка, ..., элемент списка⟩

Конструкции фортрана	Примеры
Программа	
PROGRAM ⟨имя программы⟩ {невыполняемый оператор} {выполняемый оператор} END	PROGRAM TRAP EXTERNAL FI CALL S[(...)] END
Невыполняемые операторы	
INTEGER ⟨список⟩	INTEGER K(15),N
REAL ⟨список⟩	REAL X,Y(10)
CHARACTER [*⟨длина⟩]	CHARACTER *60 WOR(16)
LOGICAL ⟨список⟩	LOGICAL BOOL
DOUBLE PRECISION ⟨список⟩	DOUBLE PRECISION Z(8),W
COMPLEX ⟨список⟩	COMPLEX Z(6),U(8)
IMPLICIT ⟨список⟩	IMPLICIT REAL(A−Z)
PARAMETER (⟨список описания⟩)	PARAMETER (E=2.7)
DIMENSION ⟨список описания⟩	DIMENSION K(15),Y(10)
EQUIVALENCE ⟨список эквивалент.⟩	EQUIVALENCE (A,B,C)
EXTERNAL ⟨список имен⟩	EXTERNAL FI,A1A0
INTRINSIC ⟨список имен⟩	INTRINSIC DSIN
DATA ⟨список имен⟩/⟨список констан⟩/ {[,]⟨список имен⟩/⟨список конст.⟩/}	DATA M,N/1,3/X/0.1/
⟨метка⟩ FORMAT (⟨список спецификаций⟩)	10 FORMAT (F6.2,I3)

Выполняемые операторы

⟨имя⟩=⟨выражение⟩	Y=A * SQRT(B)+0.3
PRINT *,⟨список вывода⟩	PRINT *,M,X
READ *,⟨список ввода⟩	READ *,C
PRINT ⟨метка формата⟩,⟨список вывода⟩	PRINT 10,X,Y
READ ⟨метка формата⟩,⟨список ввода⟩	READ 5,R,T,I
IF (⟨логич. выр.⟩) THEN	IF (M.EQ.1) THEN
{⟨выполняемый оператор⟩}	X=Y*Y−2.
ELSE IF (⟨логич. выр.⟩) THEN	ELSE IF (M.EQ.2) THEN
{⟨выполняемый оператор⟩}	X=SIN(Y)
[ELSE]	ELSE
{{⟨выполняемый оператор⟩}}	X=COS(Y)+1.
ENDIF	ENDIF
DO ⟨метка⟩ [,] ⟨диапазон индекса⟩	DO 1,I=4*K,ABS(A),−1
{⟨выполняемый оператор⟩}	X(I)=X(I)+1.
⟨метка⟩ CONTINUE	1 CONTINUE
CALL ⟨имя подпрогр⟩[(⟨список факт. пар.⟩)]	CALL SIM(X,0.5)
GO TO ⟨метка⟩	GO TO 10
IF (⟨логич. выр.⟩) ⟨выполн. оператор⟩	IF (M.GT.1) X=X+1
IF (⟨арифм. выр.⟩) ⟨метка⟩,⟨метка⟩,⟨метка⟩	IF (W) 1,10,100
ASSIGN ⟨метка⟩ TO ⟨имя⟩	ASSIGN 10 TO K
GO TO ⟨имя⟩	GO TO K

Процедуры

SUBROUTINE ⟨имя⟩ [(⟨список форм. пар.⟩)]	SUBROUTINE SIM(A,B)
{⟨невыполняемый оператор⟩}	REAL A,B
{⟨выполняемый оператор⟩}	IF (B.LT.0.) RETURN
END	END
[⟨тип⟩] FUNCTION ⟨имя⟩ [(⟨список форм. пар.⟩)]	REAL FUNCTION S(A) REAL A
{⟨невыполняемый оператор⟩}	S=ABS(COS(A))
{⟨выполняемый оператор⟩}	RETURN
END	END
ENTRY ⟨имя входа⟩ [(⟨список форм. пар⟩)]	ENTRY V(X,Y)
SAVE [⟨список локальных переменных⟩]	SAVE /TOR/,I

Операторы ввода — вывода с файлами

WRITE (⟨управляющий список⟩) ⟨список⟩	WRITE (3,1) Y,(X(I),I=9,1,−2)
READ (⟨управляющий список⟩) ⟨список⟩	READ (N,'(A)') K
OPEN (⟨устр-во⟩,⟨список открытия⟩)	OPEN (3,FILE='DATA')
CLOSE (⟨устр-во⟩,⟨список закрытия⟩)	CLOSE (3)
INQUIRE (⟨устр-во⟩,⟨список опроса⟩)	INQUIRE (2,EXIST=LOG)

INQUIRE (⟨файл⟩, ⟨список опроса⟩)	INQUIRE (VAP,OPENED=L)
BACKSPACE ⟨устройство⟩	BACKSPACE 5
ENDFILE ⟨устройство⟩	ENDFILE 3

Управление вводом—выводом файла

Элементы ⟨управляющего списка⟩ ввода—вывода	[UNIT=] ⟨целое выражение⟩ [FMT=] ⟨текстовое выражение⟩ или [FMT=] ⟨форматная метка⟩ REC=⟨целое выражение⟩ END=⟨метка⟩ IOSTAT=⟨имя целой переменной⟩
Элементы ⟨списка открытия⟩	[UNIT=] ⟨целое выражение⟩ FILE=⟨текстовое выражение⟩ ACCESS=⟨текстовое выражение⟩ STATUS=⟨текстовое выражение⟩ FORM=⟨текстовое выражение⟩ RECL=⟨целое выражение⟩ BLANK=⟨текстовое выражение⟩ ERR=⟨метка⟩ IOSTAT=⟨имя целой переменной⟩
Элементы ⟨списка закрытия⟩	[UNIT=] ⟨целое выражение⟩ STATUS=⟨текстовое выражение⟩ ERR=⟨метка⟩ IOSTAT=⟨имя целой переменной⟩
Элементы ⟨списка опроса⟩	[UNIT=] ⟨целое выражение⟩ FILE=⟨текстовое выражение⟩ EXIST=⟨имя логической переменной⟩ OPENED=⟨имя логической переменной⟩ NUMBER=⟨имя целой переменной⟩ NAMED=⟨имя логической переменной⟩ NAME=⟨имя текстовой переменной⟩ ACCESS=⟨имя текстовой переменной⟩ SEQUENTIAL=⟨имя логич. переменной⟩ DIRECT=⟨имя логической переменной⟩ FORM=⟨имя текстовой переменной⟩ FORMATTED=⟨имя логической переменной⟩ UNFORMATTED=⟨имя логич. переменной⟩ RECL=⟨имя целой переменной⟩ NEXTREC=⟨имя целой переменной⟩ BLANK=⟨имя текстовой переменной⟩ ERR=⟨метка⟩ IOSTAT=⟨имя целой переменной⟩

⟨список спецификаций формата⟩

необязательные элементы

повторяемые	I⟨кол-во позиций⟩ 4I5
	I⟨кол-во позиций⟩.⟨цифры⟩ I6.3
	A⟨кол-во позиций⟩ A6
	L⟨кол-во позиций⟩ L2
	F⟨кол-во позиций⟩.⟨цифры⟩ F6.3
	E⟨кол-во позиций⟩.⟨цифры⟩ E13.6
	D⟨кол-во позиций⟩.⟨цифры⟩ D10.5
	G⟨кол-во позиций⟩.⟨цифры⟩ G10.3
неповторяемые	⟨масштабный множитель⟩P 3PE13.6
	⟨текстовая константа⟩ 'VARIANT'
	⟨холлеритова константа⟩ 7HVARIANT
	T⟨номер позиции⟩ T8
	TL⟨число позиций⟩ TL5
	TR⟨число позиций⟩ TR6
	⟨число позиций⟩ X 3X
	S
	SP
	SS
	BN
	BZ
	/
	:

Определение глобальных данных

COMMON [/имя общего блока/] ⟨список⟩	COMMON /STAR/X,Y(10)
BLOCK DATA [⟨имя блока данных⟩] {⟨невыполняемые операторы⟩}	BLOCK DATA DEF COMMON /S/I,J,K
END	DATA I,J,K/3,4,5/ END

Операторы возврата, паузы, останова

RETURN [⟨целое выражение⟩]	RETURN 1
PAUSE [...]	PAUSE 5
STOP	

Прочие конструкции

⟨имя⟩ 1—6 букв и/или десят. цифр, начинается с буквы

⟨метка⟩ последовательность 1—5 десят. цифр

⟨целое выражение⟩	выражение, содержащее арифм. операции над целыми
⟨арифм. выражение⟩	выражение, содержащее арифм. операции над числами
арифметические операции	+ сложение
	− вычитание
	* умножение
	/ деление
	** возведение в степень
текстовая операция	// объединение

подцепочка	⟨имя текстовой переменной⟩(⟨целое выражение⟩:⟨целое выражение⟩)
операции отношения	.EQ. равно
	.NE. не равно
	.LT. меньше чем
	.GT. больше чем
	.LE. меньше или равно
	.GE. больше или равно
логические операции	.NOT. отрицание
	.AND. истина, если оба — истина
	.OR. истина, если один или оба истина
	.EQV. истина, если значения операндов одинаковы
	.NEQV. истина, если значения операндов различны
Логические константы	.TRUE.
	.FALSE.
Числовые константы	целая
	вещественная
	двойной точности
	комплексная
текстовая константа	'⟨последовательность символов⟩'
холлеритова константа	⟨целое⟩H⟨последовательность символов⟩
	(⟨целое⟩ — длина последовательности)
⟨описание массива⟩	⟨имя⟩(⟨список границ размерности⟩)
⟨границы размерности⟩	⟨нижняя граница⟩:⟨верхняя граница⟩
⟨элемент массива⟩	⟨имя⟩(⟨список индексов⟩)
⟨неявное описание⟩	⟨тип⟩(⟨буквы, диапазон букв⟩)
⟨диапазон индекса⟩	⟨имя целой перем.⟩=⟨целое выражен.⟩,
	⟨целое выражение⟩[,⟨целое выражение⟩]
⟨неявный цикл⟩	(⟨список выражений⟩,⟨диапазон индекса⟩)
⟨эквивалентность⟩	(⟨список имен⟩)

ЛИТЕРАТУРА

1. *Ахо А., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов. М., 1979.

2. *Бахвалов Н. С., Жидков Н. П., Кобельков Г. М.* Численные методы. М., 1987.

3. *Баяковский Ю. М., Галактионов В. А., Михайлова Т. Н.* Графор. Графическое расширение фортрана. М., 1985.

4. *Березин И. С., Жидков Н. П.* Методы вычислений. М., 1959. Т. 1. 1960. Т. 2.

5. *Бирюков В. В., Рыбаков А. В., Шакула Ю. П.* Введение в систему программирования ОС РВ. М., 1986.

6. *Боровин Г. К., Комаров М. М., Ярошевский В. С.* Ошибки-ловушки при программировании на фортране. М., 1987.

7. *Валикова Л. И., Вигдорчик Г. В., Воробьев А. Ю., Лукин А. А.* Операционная система СМ ЭВМ РАФОС. М., 1984.

8. *Волков Е. А.* Численные методы. М., 1982.

9. *Де Бор К.* Практическое руководство по сплайнам. М., 1985.

10. *Калиткин Н. Н.* Численные методы. М., 1978.

11. *Камке Э.* Справочник по обыкновенным дифференциальным уравнениям. М., 1971.

12. *Кейлингерт П.* Элементы операционных систем. М., 1985.

13. *Колдербэнк В.* Программирование на фортране. М., 1986.

14. *Кристиан К.* Введение в операционную систему UNIX М., 1985.

15. *Мак-Кракен Д., Дорн У.* Численные методы и программирование на фортране. М., 1977.

16. *Марчук Г. И.* Методы вычислительной математики. М., 1989.

17. *Меткалф М.* Оптимизация в фортране. М., 1985.

18. *Митчел Э., Уэйт Р.* Метод конечных элементов для уравнений с частными производными. М., 1981.

19. *Найфэ А.* Введение в методы возмущений. М., 1984.

20. *Ортега Дж., Пул У.* Введение в численные методы решения дифференциальных уравнений. М., 1986.

21. *Прудников А. П., Брычков Ю. А., Маричев О. И.* Интегралы и ряды. Элементарные функции. М., 1981.

22. *Прудников А. П., Брычков Ю. А., Маричев О. И.* Интегралы и ряды. Специальные функции. М., 1983.

23. *Прудников А. П., Брычков Ю. А., Маричев О. И.* Интегралы и ряды. Дополнительные главы. М., 1986.

24. *Самарский А. А.* Введение в численные методы. М., 1987.

25. *Самарский А. А., Николаев Е. С.* Методы решения сеточных уравнений. М., 1978.

26. Сборник научных программ на фортране. Вып. 1. Статистика / Пер. с англ. С. Я. Виленкина. М., 1974.

27. Сборник научных программ на фортране. Вып. 2. Матричная алгебра и линейная алгебра / Пер. с англ. С. Я. Виленкина. М., 1974.

28. Системы параллельной обработки / Под ред. Д. М. Ивенса, 1985.

29. *Тихонов А. Н., Арсенин В. Я.* Методы решения некорректных задач. М., 1986.

30. *Тихонов А. Н., Гончарский А. В., Степанов В. В., Ягола А. Г.* Регуляризирующие алгоритмы и априорная информация. М., 1983.

31. *Тихонов А. Н., Самарский А. А.* Уравнения математической физики. М., 1972.

32. *Форсайт Дж., Малькольм М., Моулер К.* Машинные методы математических вычислений. М., 1980.

33. *Янке Е., Эмде Ф., Леш Ф.* Специальные функции. Формулы, графики, таблицы. М., 1977.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

Абсолютная погрешность 242
Автоматический выбор шага 395
Адрес 18
— виртуальный 19
Адресация 21
Алгол 10
Алгоритм 9, 11
— базовый 48
— QR определения собственных векторов 333
— — — — значений 333
Алфавитно-цифровое печатающее устройство (АЦПУ) 37
Алфавитно-цифровой терминал 38
Альтернативная схема 47
Анализ размерностей 172
Аналитические методы 165, 166
Англо-русский словарь терминов 525
Апостериорная оценка погрешности 326
Аппроксимация 248, 383
Априорная оценка погрешности 325
Арифметическое выражение 68
— и логическое устройство 17
Архитектура адресации 21
— алгоритма 46
— ЭВМ 16
— — на основе потока данных 26
Асимптотические разложения интегралов от быстроосциллирующих функций 228
— формулы 224
Асимптотический ряд 225
Ассемблер 10

Базовые логические схемы 46
Базовый алгоритм 48
Байт 18
Бейсик 10
Библиотечная программа 69
— функция 69
Бит 18
— операнда 20
Блок-схема 46
Буфер 19

Вариационные методы 416
Ввод — вывод 37, 39
Ввод с терминала 147
Вводной файл 145
Вектор невязки краевых условий 411
— собственный 188
Вес в квадратурной формуле 277
Вещественное n-мерное пространство 187
Виртуальный адрес 19
Влияние ошибок округления 245
Внешняя программа 59
Второе поколение ЭВМ 8
Вход в систему ОС РВ 148
Выбор ведущего элемента исключения в методе Гаусса 318
Выводной файл 145
Выполнение программы 149
Выполняемый оператор 67
Выражение арифметическое 68
— логическое 80
Вырожденная матрица 192

Выход из системы ОС РВ 148
Вычисление обратной матрицы 316
— определителя матрицы 316
— произведения двух матриц 78
— специальных функций 42
— элементарных функций 41
— —комплексного переменного 42
Вычисления параллельные 23
— с двойной точностью 131
— серийные 246
Вычислительная задача обратная 236
— —прямая 236
— математика 164
Вычислительный конвейер 24
— метод 11
— процесс 54

Глобальный минимум 365
Граница сеточной области 424

Дамп 141
Дефект сплайна 256
Диагностика 160
Диск магнитный 33
— —гибкий 34
Дискретизация непрерывной задачи 238
— обыкновенного дифференциального уравнения 375
— Ритца 420
— уравнения с частными производными вариационным методом 420
— — — — —конечными разностями 423, 433
— — — — —методом прямых 440
Диспетчер 138
Доступ последовательный 128
— прямой 128
Драйвер 140

Евклидова норма 187

Жесткое уравнение 396

Загрузчик 143
— начальный 143
Задача возмущений 308
— вычислительная обратная 236
— —прямая 236
— Дирихле 216
— интерполяции 252
— корректно поставленная 216
— Коши для линейного дифференциального уравнения второго порядка 402
— — —обыкновенного дифференциального уравнения 375
— — —системы линейных дифференциальных уравнений 408
— — — —нелинейных дифференциальных уравнений 413
— линейная 250, 375
— линейной оптимизации каноническая 335
— нелинейная 251, 375
— равномерного приближения 250
— среднеквадратичного приближения 250
Замена переменных в модели задачи 176
Значащая цифра 243
Золотое сечение 372

Идентификатор 64
Инструментальные программы 133
Интеграл Римана 277
Интегральное уравнение 195
— —вырожденное 196
Интегрирование численное 276
— функции двух переменных 297
Интерполяция 252
— линейная 92
— полиномами Лагранжа 252
— сплайнами 257
Интерпретатор языка команд 142
Искусство вычислений 172
Исправленный метод Эйлера 388
Исходный файл 149
Итерация Зейделя 326
— простая 323

Канал 139
— мультиплексный 139
Каноническая задача линейной оптимизации 335
Карта данных 52
— памяти 53
Каталог библиотеки фортран-программ 445
— файлов 141
Качественные методы 165
Квадратурная формула 278
Квант времени 139
Классификация архитектуры ЭВМ 23
— задач теории приближений 250
Код идентификации пользователя 145
Команды текстового редактора 152
Компилятор 136
Константа фортрана 64
— — вещественная 64
— — двойной точности 65
— — комплексная 65
— — логическая 66
— — текстовая 66
— — целая 64
Корректность постановки задачи 216
Коэффициент жесткости 399, 413
— Фурье 184
Критерий локализации собственных значений 194
— Чебышева 265
Кубический сплайн 258

Линеаризация 347
Линейная задача 250, 375
— оптимизация 335, 337
Линейное программирование 335
Логические выражения 80
— операции 80
— переменные 80
Локальный минимум 365

Магнитная лента 35
Мантисса числа 32
Массив 58, 67
Масштабирование 172
Математическая модель 11
Матрица вращения 332
— вырожденная 192
— невырожденная 192
— отражения 330
— положительно определенная 317
— почти треугольная 329
— разреженная 428, 429
— симметричная 317
— трехдиагональная 329
Метка 71
Метод аналитический 165, 166
— — решения обыкновенных дифференциальных уравнений 200
— — — систем линейных уравнений 190
— — — уравнений с частными производными 213
— — суммирования ряда 182
— — Фурье-анализа 183
— бисекции решения нелинейных уравнений 361
— вариационный 416
— возмущений 165, 167
— — в линейной алгебре 308
— — вычисления интегралов 227
— — регулярный 168
— — решения обыкновенных дифференциальных уравнений 231
— — сингулярный 170
— градиентного спуска 109, 368, 370
— дробных шагов 438
— Зейделя 326
— золотого сечения 372
— исключения Гаусса 312
— — с полным упорядочением 319
— итераций решения нелинейных уравнений 350
— — — систем линейных уравнений 322
— касательных 357
— качественный 165
— конечных элементов 422
— Куммера 182

— наименьших квадратов 268
— Ньютона решения нелинейных уравнений 356
— перебора 365
— подобия 213, 223
— покоординатного спуска 367
— полудискретный 440
— приближенный 312
— прогонки дискретной 114, 406
— —непрерывной 210
— прямой решения систем линейных уравнений 312
— прямых с конечно-разностной аппроксимацией 442
— Рунге—Кутта 389
— —четвертого порядка 113, 391
— сведения к интегральному уравнению 212
— стационарной фазы 229
— стрельбы 411
— точный 312
— Фурье решения уравнений с частными производными 213, 217, 219, 221
— численный 165, 170
— —теоретический 235
— Эйлера решения дифференциального уравнения исправленный 388
— — — — —явный 112, 388
— суммирования тригонометрических рядов 185
Минимум глобальный 365
— локальный 365
Мультиплексный канал 139
Мультипрограммирование 137, 138

Насыщение численного метода 293
Невыполняемый оператор 67, 87
Невырожденная матрица 192
Нелинейная задача 251, 375
— оптимизация 108, 363
Нестационарное уравнение 215
Неустойчивые схемы 386
Неявный цикл 79
Норма вектора 187
— евклидова 187
— матрицы 187
— равномерная 250
— сеточная 381
— среднеквадратичная 250
Нормализованная запись числа 32

Обратная прогонка 114
Обратный ход 314
Общая шина 29
Объектный файл 149
Оператор ввода—вывода READ—WRITE 71
— внешней процедуры EXTERNAL 87
— возврата RETURN 83
— вызова подпрограммы CALL 85
— выполняемый 67
— конца программы END 71
— начальных данных DATA 87
— невыполняемый 67, 87
— общих блоков COMMON 84, 87
— описания типа 67
— —COMPLEX 67
— —DOUBLE PRECISION 67
— —INTEGER 67
— —LOGICAL 67
— —REAL 67
— передачи управления GO TO 76
— —IF арифметический 76
— —IF логический 81
— присваивания 60
— —логический 81
— продолжения CONTINUE 77
— структурный логический 125
— формата FORMAT 72
— цикла DO 77
— эквивалентности EQUIVALENCE 87
Операторы фортрана 77, 124, 529
Операционная система 134
Операция арифметическая 68
— логическая 80
Описание библиотечной программы A1A0 451
— — —A1A1 452
— — —A1A2 452
— — —A1A3 452

— — —A1A4 452
— — —A1A5 453
— — —A1A6 453
— — —A1A7 453
— — —A1A8 454
— — —A1A9 454
— — —A1B0 454
— — —A3A0 455
— — —A3A1 456
— — —A3A2 456
— — —A4A0 458
— — —A4A1 458
— — —A4A2 459
— — —A5A0 459
— — —A5A1 460
— — —A5A2 460
— — —A6A0 462
— — —A6A1 462
— — —A7A0 463
— — —A7A1 463
— — —A7A2 464
— — —A8A0 464
— — —A8A1 465
— — —A8A2 466
— — —A8A3 466
— — —A8A4 466
— — —A8A5 467
— — —A9A0 468
— — —A9A1 469
— — —B0A0 470
— — —B0A1 470
— — —B0A2 471
— — —B0A3 471
— — —B1A0 472
— — —B2A0 474
— — —B3A0 475
— — —B4A0 475
— — —B4A1 476
— — —B4A2 477
— — —B5A0 477
— — —B6A0 479
— — —B6A1 480
— — —B7A0 481
— — —B7A1 483
— — —B8A0 484
— — —B8A1 486
— данных 58
— потока данных алгоритма 52
Оптимизация ввода — вывода 122
— линейная 335
— — в геометрической интерпретации 337
— нелинейная 108, 363
— программ 116
Организация памяти 18
Ортогональные полиномы 262
Особенности версии фортрана 77, 123
Отладка программы 137, 160
Относительная погрешность 242
Оценка погрешности квадратурной формулы 279, 281, 282
— — метода простой итерации 325
— — определения собственных векторов 308
— — — — значений 307, 308
— — решения обыкновенного дифференциального уравнения 392
— — — системы линейных уравнений 303
— — — уравнения с частными производными 429, 437
— числа операций метода Гаусса 320
Ошибка округления 245
Память ЭВМ 18
Параболический сплайн 257
Параллельные вычисления 23
Параметр фактический 82
— формальный 82
Пароль пользователя 148
Паскаль 10
Первое поколение ЭВМ 7
Переменная безразмерная 173
— логическая 80
— фортрана 66
Планировщик задач 138
Повторная схема 47
Погрешность абсолютная 242
— аппроксимации 382
— вычислений 241
— — функции 244
— модели 240

— округления 245
— относительная 242
— численного метода 240
Подпрограмма 85
Поиск минимума методом градиентного спуска 109, 368, 370
— — —золотого сечения 372
— — —перебора 365
— — —покоординатного спуска 367
— эмпирических закономерностей 249
Полином Лежандра 293
— наилучшего равномерного приближения 251
— —среднеквадратичного дискретного приближения 251
— наименее уклоняющийся от нуля 264
— Чебышева 261
Полудискретный метод 440
Пользователь 15
Порядок числа 32
Последовательная схема 47
Построение разностной схемы 383, 403, 423
Построитель задач ОС РВ 157
Почти треугольная матрица 329
Правило Крамера 167
— Рунге 290
Предвычисления 246
Представление букв и символов в памяти 32
Прерывание 139
Приближение равномерное 261
— среднеквадратичное 268
— —дискретное 271
— —интегральное 268
Приведение матрицы к почти треугольному виду 329
Прикладные программы 133
Применение библиотечной программы A3A0 256
— — —A3A1 259
— — —A3A2 274
— — —A4A0 292
— — —A4A1 296
— — —A4A2 299
— — —A8A5 321
— — —A9A0 321
— — —B0A0 334
— — —B0A3 335
— — —B1A0 343
— — —B4A0 363
— — —B4A1 359
— — —B4A2 360
— — —B5A0 373
— — —B5A1 370
— — —B6A0 396
— — —B6A1 401
— — —B7A0 412
— — —B7A1 414
— — —B8A0 431
— — —B8A1 443
— рядов Тейлора 387
Пример некорректно поставленной задачи 216
Примеры задач линейной оптимизации 336
— методов аналитических 166
— —возмущений 167
— —качественных 170
— —численных 165
Принцип максимума 426
Прогонка обратная 114
— прямая 114
Программа 10, 11
— библиотечная 69
— внешняя 59
— инструментальная 133
— компилятор 136
— модульной структуры 86
— прикладная 133
— работы с файлами 158
— редактор 135
— системная 133
— транслятор 135, 136
— элементарная 90
— —аппроксимации 92
— —градиентного спуска 109
— —золотого сечения 108
— —интегрирования 94
— —интерполяции 92
— —матричной алгебры 102
— —метода бисекции 107
— — —прогонки 114

— — —Рунге—Кутта 113
— — —Эйлера 112
— —решения интегрального уравнения 105
— — —краевой задачи 115
— — —нелинейного уравнения 106
— — —системы линейных уравнений 104
— —суммирования рядов 96
— —табулирования 91
— —Фурье-анализа 99
— —численного дифференцирования 101
Программирование 54
— структурное 58
Проектирование алгоритма 48
Процедура 55
Процесс 54
— вычислительный 54
— экономизации 267
Прямая прогонка 114
Прямой ход 313
Пятое поколение ЭВМ 8

Работа на ЭВМ 12
Равномерное приближение 261
Равномерность асимптотических формул 226
Раздел памяти 137
Разложение асимптотическое 225
— —интегралов 227
— матриц на треугольные множители 317
— *LU* матриц 315
— *QR* матриц 332
Размерность массива 67
Разностная схема 383
Разреженная матрица 428, 429
Реализация численного метода 235
Регистр 18, 29
Регулярный метод возмущений 168
Редактор 135, 151
— связей 136
Ресурс ЭВМ 16
Решение разностных уравнений 427

Свертка функций 196
Семейство вычислительных алгоритмов 41
Серийные вычисления 246
Сетка 381
Сеточная норма 381
— область 423
Сжимающее отображение 349
Символ *O* большое 224
— *o* малое 224
Симметричная матрица 317
Симплекс-метод 340
Сингулярный метод возмущений 170
Система ограничений канонической задачи линейного программирования 335, 336
— операционная 134
— реального времени 143, 144
— счисления 30
— —восьмеричная 31
— —двоичная 31
— —десятичная 31
— управления базой данных 133
— файлов 140
Системные программы 133
Слабое решение краевой задачи 420
Собственная функция 218
Собственное значение 188, 218
Собственный вектор 188
Советы по применению курса 14
Спецификация 72
— файла 145
Сплайн 251, 256
— кубический 258
— параболический 257
Справочник файлов 141
Сравнение линейных и нелинейных уравнений 344
Среднеквадратичная норма 250
Среднеквадратичное приближение 268
Стационарное уравнение 215
Структура программы 70
Структурная алгоритмизация 52
Структурное программирование 58
Схема альтернативная 47
— базовая логическая 46

— Горнера 84
— Кранка — Николсона 436
— повторная 47
— последовательная 47
— разностная 383
— — для двумерного уравнения теплопроводности 437
— — — уравнений с частными производными нестационарных 433
— — — — — — — стационарных 422
— — задачи Коши для обыкновенного дифференциального уравнения 383
— — краевой задачи для обыкновенного дифференциального уравнения 402
— — неявная 384
— — явная 384
— Эйткена 255
Сходимость разностной схемы 383, 385

Теорема Пуанкаре 232
— Чебышева 264
Тип данных 67
Типы погрешностей 240
Точка чебышевского альтернанса 265
Точность аппроксимации 382
Транслятор 135, 136
Третье поколение ЭВМ 8
Трехдиагональная матрица 329

Узел в квадратурной формуле 277
— интерполяции 252
— сетки 381, 423
— — внутренний 423
— — граничный 424
— — соседний 423
Улучшение структуры циклов 120
Унимодальная функция 371
Управление памятью 137
— потоком данных 26
— процессором 138
— устройствами 139
— файлами 140
Уравнение волновое 213
— Гельмгольца 423
— жесткое 396
— интегральное 195
— Лапласа 216
— нестационарное 215
— обыкновенное дифференциальное 375
— Пуассона 216
— разностное трехточечное 405
— стационарное 215
— с частными производными 115
— теплопроводности двумерное 437
— — одномерное 214, 435
Ускорение на передачах управления 121
Устойчивость прогонки 407
— разностной схемы 383, 385
Устройство арифметическое и логическое 17
— управления 17

Файл 36
— вводной 145
— выводной 145
— готовый к исполнению 149
— исходный 149
— листинга 156
— объектный 149
— с последовательным доступом 128
— — прямым доступом 128
Формула асимптотическая 224
— квадратурная 278
— — Гаусса 278
— — — для квадрата 297
— — простейшая 279
— — прямоугольников 279
— — составная 282
— — с переменным шагом 286
— — — постоянным шагом 282
— — Симпсона 281
— — точная 278
— — трапеций 281
— Фробениуса 191

— численного дифференцирования 382
— —интегрирования 228
Фортран 10, 61
— 66 61
— 77 62
— 8х 62
Функционал 416
Функция библиотечная 69
— комплексного переменного 180
— специальная 180
— унимодальная 371
— целевая 336, 363
— элементарная 178
Функция-подпрограмма 83
Функция-формула 82
Фурье-анализ 43, 99, 183

Ход обратный 314
— прямой 313

Целевая функция 336, 363
Центральный процессор 20, 29
Цикл 77
— неявный 79
Цифра значащая 243
— —верная 243

Четвертое поколение ЭВМ 8
Численное дифференцирование 382
— интегрирование 276
Численные методы 165
Число верных значащих цифр 243
— нормализованное 32
— обусловленности 304
— —матрицы 305
— —системы 304
— Рейнольдса 176

Шаблон 428
Шаг сетки 381
— симплекс-метода 341
Шар нормированного пространства 302

Экономизация степенных рядов 266
Экстраполяция Ричардсона 292
Элементарные программы 90
— функции 178
Этапы дискретизации 375

Явный метод Эйлера 388
Ядро операционной системы 137
Язык команд системы 142
— программирования 10

Учебное издание

Боглаев Юрий Павлович

ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ПРОГРАММИРОВАНИЕ

Зав. редакцией Е. С. Гридасова. Редактор Ж. И. Яковлева.
Художественный редактор В. И. Пономаренко. Технический редактор Н. В. Яшукова.
Корректор Г. И. Кострикова

ИБ № 8155

Изд. № ФМ-971. Сдано в набор 04.12.89. Подп. в печать 04.07.90. Формат 60×90/16. Бум. офс. № 2. Гарнитура таймс. Печать офсетная. Объем 34,0 усл. печ. л. 34,0 усл. кр.-отт. 33,52 уч.-изд. л. Тираж 50000 экз. Зак. № 3519. Цена 1 р. 30 к.

Издательство «Высшая школа», 101430, Москва, ГСП-4, Неглинная ул., д. 29/14

Ордена Октябрьской Революции и ордена Трудового Красного Знамени МПО «Первая Образцовая типография» Государственного комитета СССР по печати. 113045, Москва, Валовая, 28